UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

SISTEMA DE EVALUACIÓN AUTOMÁTICA DE TAREAS DE PROGRAMACIÓN MEDIANTE ANÁLISIS DE CÓDIGO

Daniel Jimeno Ortiz

Tutor: Francisco Jurado Monroy

Ponente: Jaime Moreno Llorena

Julio de 2018

SISTEMA DE EVALUACIÓN AUTOMÁTICA DE TAREAS DE PROGRAMACIÓN MEDIANTE TÉCNICAS DE ANÁLISIS DE CÓDIGO

Autor: Daniel Jimeno Ortiz
Tutor: Francisco Jurado Monroy
Ponente: Jaime Moreno Llorena

Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Julio de 2018

AGRADECIMIENTOS

El comienzo de mi largo, a veces corto, camino tiene lugar en Septiembre del año 2013; nueva etapa, nuevas amistades y mi primer contacto con la informática. Nadie me dijo que sería fácil, pero aun así sabía que era el camino que quería seguir y al que quería dedicar mi tiempo y mi futuro.

Se quedan en mi mente los malos y los buenos momentos, pero sobretodo los buenos. Esa satisfacción al conseguir que las prácticas funcionen o al ver un aprobado a la derecha de mi nombre, no me la quita nadie. Las largas noches programando siempre serán recordadas, por lo menos tenía buena compañía, aunque fuese por videollamada. La cuenta atrás a la espera de la llegada de los exámenes y los nervios esperando para entrar a hacerlos, los descansos en la cafetería que acababan siendo repasos y conversaciones acerca de las prácticas, o incluso, la manía de contarle a todo el mundo mis preocupaciones, aunque nadie entendiese de que estaba hablando. Todas estas experiencias pueden parecer un auténtico dolor de cabeza, y no digo que no lo hayan sido, pero las he disfrutado y han conseguido que hoy esté aquí, que hoy sea capaz de decir que estoy orgulloso de mi mismo y que hoy pueda decir y demostrar que sé un poquito más de informática.

Este camino tan largo, que ahora veo tan corto, no lo habría podido recorrer sin el apoyo de todos mis seres queridos o sin el apoyo y ayuda de mis compañeros y de mis profesores. A todos y cada uno de vosotros, os debo algo, a algunos más que a otros, pero todos merecéis que os dé las gracias. Gracias por haberme dado fuerzas para seguir adelante y esforzarme al máximo para conseguir mis objetivos, gracias por enseñarme que siempre puedo dar un poquito más de mí mismo y gracias sobre todo por estar a mi lado, sufriendo conmigo, en todo momento.

Esta etapa que se cierra siempre quedará grabada en mi mente y en mi corazón, con todas y cada una de mis vivencias, las de estudio y las de ocio, pues siempre ha habido algún espacio para pasarlo bien y disfrutar de una buena fiesta, pero sobretodo me quedaré con vosotros y cada una de nuestras experiencias.

Gracias por tanto a todos los que sintáis que formáis parte de estas palabras.

RESUMEN

Debido al crecimiento exponencial del número de individuos que forman parte de los equipos de desarrollo de aplicaciones, es cada vez más difícil controlar, verificar y corregir los códigos que se escriben día a día en una empresa, una universidad o más abiertamente en el mundo. Un ejemplo claro de éste problema sería el código que tendría que evaluar el supervisor de tres programadores de una empresa o el código de los alumnos de una clase que tendría que corregir un profesor de universidad.

Por lo tanto, surge la necesidad de crear y desarrollar herramientas que sean capaces de realizar los controles correspondientes al código deseado y de facilitar el trabajo de aseguramiento de calidad y de uso de todos los programas informáticos que se crean, amplían o modifican día a día en el mundo. Estas herramientas, por consiguiente, estarán basadas en el análisis de código que será necesario realizar sobre todos los datos que quieran ser monitorizados.

En relación con las necesidades mencionadas anteriormente que surgen en las universidades para que un solo profesor sea capaz de evaluar todas las prácticas que realiza el grupo de alumnos al que éste dicta clase, se ha desarrollado ***CheckOode***, una herramienta que se encarga de realizar un análisis de datos suficientemente completo como para agilizar y mejorar en un gran porcentaje el proceso de corrección de las prácticas de la asignatura de Proyecto de Programación de primer curso del Grado en Ingeniería Informática de la Universidad Autónoma de Madrid. ***CheckOode*** es una herramienta que es capaz de autoevaluar el código que han entregado los alumnos y posteriormente, mostrar todos los resultados de dicho análisis, en una interfaz web local, de manera que la visualización de los mismos sea intuitiva y muy amplia en contenidos para facilitar y mejorar el trabajo del profesor.

Además, ***CheckOode*** permite al usuario, en este caso el profesor, interactuar con los resultados obtenidos y realizar el análisis de varios proyectos simultáneamente, de manera que a la hora de visualizar los datos y corregir las prácticas de los alumnos pueda hacerlo seguidamente, sin tener que realizar uno a uno el análisis de cada una de las prácticas a evaluar.

CheckOode se presenta como una solución real a los problemas que surgen hoy en día para la evaluación de códigos de programación debido al gran número de los mismos y a la cantidad de herramientas que nos permiten que su crecimiento sea exponencial y en algunos casos incluso sea generado automáticamente.

PALABRAS CLAVE

Análisis de Código, Interfaz de Usuario, Interfaz Web, Bash Script, Linux, Navegador

ABSTRACT

Due to the exponential growth of the number of developers, it is increasingly difficult to control, verify and correct the codes that are written every day in a company, a university or more openly in the world. A clear example of this problem would be the code that a supervisor of three programmers of a company would have to evaluate or the code of the students of a class that a university professor would have to correct.

Therefore, this need arises to create and develop tools that are capable to perform the controls corresponding to the desired code and to facilitate the work of quality assurance and use of all the computer programs that are created, expanded or modified every day in the world. These tools, therefore, will be based on the analysis of the code that will be necessary to perform on all the data that wants to be monitored.

In relation to the needs mentioned above, that arise in universities for a single teacher to be able to evaluate all the practices carried out by the group of students from his class, CheckOode has been developed. CheckOode is a tool that is responsible to carry out a sufficiently complete data analysis to speed up and improve in a large percentage the process of correcting the practices of the first year subject Proyecto de Programación of the Degree in Computer Engineering of the Universidad Autónoma de Madrid. CheckOode is a tool that is able to self-assess the code that students have delivered and then show all the results of that analysis in a local web interface so that the visualization of this content is intuitive and very broad in content to facilitate and improve the teacher's work.

In addition, CheckOode allows the user, in this case the teacher, to interact with the results obtained and perform the analysis of several projects simultaneously, so that when viewing the data and correct the practices of students, he can do it without having to perform one by one the analysis of each of the practices to be evaluated.

CheckOode is presented as a real solution to the problems that arise today for the evaluation of programming codes due to the large number of them and the number of tools that allow us to grow exponentially and in some cases even be generated automatically.

KEYWORDS

Code Analysis, User Interface, Web Interface, Bash Script, Linux, Browser

Índice

Índice de Figuras

Índice de Tablas

Índice de Figuras Anexos

1. Introducción

1.1 Marco del Proyecto

En las asignaturas de programación y basadas en proyectos, el tiempo del que dispone cada profesor con respecto al número de proyectos y prácticas que le corresponde corregir es limitado, y esto implica que la tarea de evaluar de manera detallada los diferentes componentes software desarrollados por los alumnos para cada uno de los proyectos sea una labor ardua y costosa en tiempo.

CheckOode es una herramienta que, haciendo uso de algunas herramientas ya existentes para el análisis de código, impulsa y mejora los resultados de dichas herramientas y las utiliza para desarrollar un programa que sea capaz de analizar los proyectos que han creado los alumnos de la asignatura de Proyecto de Programación de primer curso de Grado en Ingeniería Informática de la Universidad Autónoma de Madrid. El programa se encarga de realizar un análisis completo de dichos proyectos y posteriormente mostrar todos los resultados, al profesor, mediante una interfaz web.

1.2 Motivación

La difícil tarea de tomar la decisión de cuál debe ser el tema sobre el que realizar el Trabajo de Fin de Grado (TFG) implica realizar una reflexión lenta y completa, y además realizar también una valoración exhaustiva de los temas propuestos por los profesores o incluso por el resto de alumnos, ya que una vez tomada la decisión el alumno debe comprometerse con el proyecto y debe dedicar mucho tiempo a su realización.

La motivación de este proyecto se basa en la existencia de un afán de cambio, por parte del estudiante, en la forma de enfocar la informática y de tratar los datos, muy importante. Es decir, la mayoría de los informáticos pasan horas, días, semanas e incluso meses seguidos analizando, planificando, diseñando e implementando proyectos diferentes que residen en el desarrollo de un código que produce un producto final, ya sea para un cliente, para sí mismos o, en el caso educativo, como una práctica para el aprendizaje propio. Entonces, ¿por qué no darle un giro a todo esto y desarrollar código en beneficio para el propio código?, ¿por qué no, en lugar de trabajar para un cliente, trabajar para nuestros colegas y para nosotros mismos permitiéndonos obtener mejores resultados? Es por esto que se toma la decisión de emprender un camino nuevo, interesante y por el que no demasiados tienen interés.

El análisis de código es un concepto muy importante en el desarrollo de software, y poder tener herramientas que faciliten estas tareas de análisis, que sirven incluso para realizar la fase de pruebas de un proyecto completo, es un nuevo camino al progreso, que enfocándose de la manera correcta puede llevarnos incluso, no solo a la existencia de herramientas de análisis de código, sino también a la existencia de herramientas, que en base al análisis de código realizado, sean capaces de hacer correcciones automáticas en el mismo, siendo capaces de evitar errores y de obtener códigos prácticamente perfectos.

1.3 Objetivos

El objetivo principal de este proyecto es diseñar e implementar una herramienta de análisis de código para la asignatura de Proyecto de Programación de primer curso del Grado en Ingeniería Informática de la Universidad Autónoma de Madrid, que sea capaz de facilitar el trabajo de corrección de las prácticas a los profesores y que en vista al futuro pueda ser ampliable para compartir su uso con los propios alumnos.

CheckOode es la herramienta que se plantea para dar solución a estos objetivos. Esta herramienta deberá ser capaz de realizar un análisis de la estructura de los proyectos entregados, de los ficheros que se encuentran en ellos, de los lenguajes utilizados para la programación de dichos proyectos, del propio estilo de programación de los mismos e incluso de algunos datos estadísticos del código. Posteriormente, la herramienta deberá ser también capaz de representar todos los datos obtenidos del análisis en una interfaz web de manera que la visualización de los datos por parte del usuario, en este caso los profesores, sea clara e intuitiva, permitiendo que estos sean capaces de evaluar los proyectos de mejor manera y permitiéndoles obtener un mejor feedback a compartir con los alumnos.

1.4 Estructura del Documento

Este documento está estructurado en apartados que tratan diferentes puntos importantes de las fases del proyecto que se ha desarrollado, incluyendo también información correspondiente al análisis de código cuyo conocimiento ha sido necesario para poder llevar a cabo el proyecto.

Para comenzar, en el apartado de *Estado del Arte sobre Análisis de Código* se puede encontrar toda la información correspondiente a los tipos existentes de análisis de código junto con las metodologías que se pueden utilizar para realizarlos y una introducción a la situación actual en la que se encuentra el análisis de código, cuáles son sus usos habituales y quien lo utiliza. Además, se mencionan y explican algunas de las herramientas existentes para el análisis de código, mencionando también cuál es su utilidad principal.

A continuación, se encuentra el apartado de *Objetivos y Funcionalidades del Proyecto* que contiene la información correspondiente a las funcionalidades que ha de cumplir el producto desarrollado y cuáles han sido los objetivos que se esperaban de la realización del proyecto.

Más adelante, comienza el apartado *Definición del Proyecto* en el que se especifica la metodología que se ha seguido para la implementación del proyecto, así como la planificación que se ha seguido para la realización del mismo, incluyendo también un Diagrama de Gantt orientativo para dicha planificación. También se especifican en el apartado las herramientas que se han utilizado para la implementación del proyecto y cuál es la funcionalidad que se le ha dado a cada una de ellas. Finalmente, se incluyen también de manera explicativa, los lenguajes de programación que se han utilizado para el desarrollo del producto.

El siguiente punto del documento trata la fase de *Análisis del Proyecto*, en el que se introduce el análisis que ha sido llevado a cabo en la etapa inicial del proyecto,

introduciendo los roles de usuario, explicando los casos de uso para dichos usuarios, también mediante propios diagramas de casos de uso y finalmente, mencionando los requisitos que se espera que cumpla el proyecto, refiriéndose tanto a los requisitos funcionales como a los requisitos no funcionales.

Siguiendo, se encuentran los puntos correspondientes al *Diseño e Implementación* del Proyecto en el cual se explica la arquitectura seguida para el desarrollo del mismo y de cada uno de sus componentes, así como también el diseño de la interfaz del proyecto y de los distintos componentes de la misma, su funcionalidad y su objetivo.

Llegando al final del documento se encuentra el apartado destinado a las *Pruebas*, en el que se explica cuáles de ellas se han realizado y la metodología seguida para realizarlas así como el alcance de las mismas.

Finalmente y como apartado final del documento se encuentra el apartado destinado a las *Conclusiones* y *Trabajo Futuro*. En este apartado se mencionan las conclusiones del documento y se explican cuáles serían las líneas futuras del proyecto, como debería ser su mantenimiento o incluso su ampliación y mejora.

2. Estado del Arte sobre Análisis de Código

2.1 Introducción

La realización de este Trabajo de Fin de Grado tiene una fase inicial y principal que consiste en realizar un primer contacto con el estado del arte sobre Análisis de Código, los tipos de análisis que podemos encontrarnos y en qué consisten. Además, es importante tener un conocimiento previo de cuál es el estado actual del Análisis de Código y cuáles son las herramientas existentes actualmente para realizarlo.

2.2 El Análisis de Código

El análisis de código se basa en cualquier proceso de análisis de información del código fuente de un programa o también de la ejecución del mismo utilizando o bien herramientas automáticas o bien realizándose manualmente por una persona.

En cuanto al código que se analiza, el código fuente, es cualquier texto que sea legible por un ser humano y que pueda ser compilado o interpretado por el correspondiente compilador o intérprete para acabar siendo ejecutado por un ordenador. Pero el análisis de código no solo engloba la evaluación del código fuente estática, sino que también se puede realizar sobre la ejecución de dicho código fuente, de manera dinámica, obteniendo otros datos de análisis diferente. En los próximos apartados se distinguen y explican a fondo dichos tipos de análisis de código. [17] [20]

Es importante mencionar que el análisis de código debe utilizarse como una herramienta de ayuda al desarrollo y nunca como un método de corrección automático del todo fiable y cuya finalidad sea exacta. Es decir, el análisis de código debe ser una ayuda a los desarrolladores para obtener un software de mejor calidad y que a la hora de utilizarlo tenga una menor probabilidad de fallo gracias a los análisis realizados.

2.2.1 Tipos de Análisis de Código

En este apartado se explican de manera más detallada los tipos principales de análisis de código que existen y se utilizan hoy en día en el mundo de la computación. Estos tipos de análisis de código son los siguientes: [1] [19]

- **Análisis de Código Estático**

  El análisis estático de código consiste básicamente en el proceso de evaluar el código fuente de un programa sin ejecutarlo. Es por ello, que se aplica directamente al código fuente sin realizar, en el mismo, ningún cambio o modificación para realizar dicho análisis.

  Lo que se busca mediante el análisis estático de código es que el analizador reciba como entrada el código fuente a analizar, lo analice y devuelva, posteriormente, unos resultados de dicho análisis que sean capaces de advertir

sobre posibles errores que se hayan encontrado en el código y mostrar sugerencias para repararlos.

Figura 2.1: *Análisis Estático*

El análisis estático de código tiene como finalidad encontrar partes de código que puedan reducir el rendimiento, tener demasiada complejidad, producir problemas de seguridad o provocar errores generales en el software [27].

Existen diferentes etapas a tener en cuenta en el análisis estático de código:

- ***Análisis del Control de Flujo***
  Esta etapa comprueba los ciclos con múltiples puntos de entrada y de salida.

- ***Análisis de Uso de Datos***
  Esta etapa detecta variables no inicializadas, repetidas o no utilizadas.

- ***Análisis de Interfaz***
  Esta etapa comprueba la consistencia de la rutina, las declaraciones del procedimiento y su uso.

Una visión algo más interiorizada del análisis de código estático sería dividirlo en tipos internos del propio análisis estático de código. Por lo tanto, podemos distinguir en dos tipos de análisis de código estáticos [24]:

- ***Análisis Estático Liviano***
  Este tipo de análisis estático de código tiene como finalidad la capacidad de evaluación de grandes cantidades de líneas de código en poco tiempo devolviendo como resultado un análisis básico pero completo de métricas relativamente simples como son por ejemplo el mal uso de construcciones dentro del programa o una estructuración incorrecta del mismo.

- ***Análisis Estático Extendido***
  Este tipo de análisis estático de código tiene como finalidad realizar un análisis más delicado del código aplicando teoremas específicos del

análisis de código, que resultan más costosos, de manera que sean capaces de demostrar propiedades del código más precisas. Este tipo de análisis estático de código necesita más recursos que el anterior.

- **Análisis de Código Dinámico**

  El análisis dinámico de código se basa en la evaluación del código durante su ejecución. Es más lento y costoso que el análisis estático pero permite observar y capturar errores que se ocultan al análisis estático, es decir, errores que ocurren únicamente en el flujo de ejecución del código.

  El análisis dinámico requiere el uso de un proceso de rastreo, que suele realizarse mediante un archivo de rastreo. Este rastreo consiste en obtener una traza del flujo de ejecución del código en la que se van analizando las diferentes partes del código y el manejo de los datos durante su ejecución. De esta manera, se consigue obtener un historial de la ejecución de un programa para que al finalizar la ejecución se obtengan los distintos resultados del flujo de dicha ejecución.

  El uso del análisis dinámico suele acabar obteniendo un análisis estadístico de las distintas métricas que se observen durante el análisis. [21]

2.2.2 Métodos de Análisis de Código

En lo que concierne a los métodos de análisis de código, existe un catálogo inicial básico de los mismos:

- **Análisis de Código Manual**

  Este método de análisis de código es realizado por una persona física, que se encarga de realizarlo mediante la comprobación visual del código y de la realización de diferentes pruebas, también manuales, en base a sus propios conocimientos en la materia.

- **Análisis de Código Automático**

  Por el contrario, el análisis de código automático es realizado utilizando herramientas específicamente implementadas para encargarse de esta tarea, siendo diseñadas para realizar un determinado tipo de análisis en base a unas instrucciones y métricas precisas y devolviendo como resultado un análisis, en la mayoría de los casos, exacto y totalmente fiable.

Existen también variedad de métodos diferentes que forman parte de los tipos mencionados anteriormente y cuyos ejemplos son algunos de los siguientes:

- Escáner de código basado en patrones
- Escáner de código basado en flujos
- Escáner de código basado en métricas
- Salida del compilador

2.3 Herramientas para el Análisis de Código

2.3.1 PDM

Herramienta de análisis de código que se encarga de detectar código duplicado, código muerto (código sin usar) y analiza la complejidad del código. Trabaja con Java, JavaScript, XSL y Ecmascript. [26] [33]

2.3.2 CheckStyle

Herramienta de análisis de código que trabaja con lenguaje de programación Java y se encarga de detectar y evaluar reglas de estilo. [22] [26]

2.3.4 SONAR

Plataforma de código abierto para gestión de la calidad del software que se obtiene de la fusión de PDM y Checkstyle y que permite analizar las métricas del código fuente y visualizar los resultados mediante informes. Trabaja con Java y ofrece soporte para C, Cobol, C#, JavaScript, PHP y Flex. [22] [23] [25] [26] [32]

2.3.5 Google CodePro Analytics

Herramienta de análisis de Google que detecta métricas, dependencias, cobertura de código y además genera test unitarios. Trabaja con Java y en particular suele estar vinculada a Eclipse. [26] [31]

2.3.6 Simian

Herramienta de análisis de código capaz de detectar código duplicado en Java, C, C#, C++, HTML y que ofrece soporte para más lenguajes. Trabaja con licencia libre y soporta proyectos opensource. [26] [30]

2.3.7 Herramientas GNU/Linux para el Análisis de Código

GNU/Linux ofrece un amplio catálogo de herramientas de análisis de Código. Si nos encontramos en un entorno basado en Debian (Debian, Ubuntu, LinuxMint, etc.) puede obtenerse un listado de algunos de los paquetes más relevantes con la instrucción:

```
apt-cache search code analysis
```

A continuación, se exponen y explican algunas de estas herramientas de análisis de código. [29]

2.3.7.1 Cppcheck

Herramienta de análisis estático de código para software escrito en lenguaje C y C++. Además, ofrece también una versión con interfaz de usuario (GUI), cppchec-gui.

Esta herramienta es capaz de detectar varios tipos de errores en el código, como por ejemplo:

- Comprobación del uso de la memoria
- Detección de referencias a punteros nulos
- Detección de variables no inicializadas
- Advertencia por detección de código muerto o redundante
- Detección de códigos peligrosos que pueden dar lugar a errores.
- Advertencia de uso de funciones obsoletas o inseguras

2.3.7.2 Frama-c

Framework dedicado al análisis de código de software escrito en lenguaje C que ofrece también una versión sin interfaz de usuario (GUI).

Fragma está basado en CIL para generar un árbol de sintaxis abstracta que soporta anotaciones escritas en lenguaje de especificación C ANSI (ACSL).

Finalmente, Fragma ofrece gran variedad de plugins con diferentes funcionalidades. Algunos de ellos son:

- ***Value Analysis.*** Computa un valor o un conjunto de valores para cada variable de un programa.
- ***Jessie.*** Verifica propiedades de manera deductiva.
- ***Impact Analysis.*** Resalta el impacto de una modificación en el código.
- ***Spare Code.*** Elimina código muerto de un programa escrito en lenguaje C.

2.3.7.3 Ruby-origami

Herramienta de análisis de PDF que sirve para parsear, modificar y generar documentos en formato PDF.

2.3.7.4 Librerías

GNU/Linux también ofrece un amplio catálogo de librerías, que se pueden incluir en un programa, que sirven para realizar análisis de código. Algunas de ellas son las siguientes:

- ***libghc-criterion-dev.*** Realiza análisis y mediciones de rendimiento.
- ***libghc-criterion-doc.*** Realiza análisis y mediciones de rendimiento y ofrece documentación.
- ***libpuma-dev.*** Realiza escáner y parseo.
- ***libpuma-doc.*** Realiza escáner y parseo de documentos.

2.4 Opiniones y Conclusiones

En conclusión, el análisis de código permite obtener una gran ayuda para los desarrolladores a la hora de la implementación de cualquier producto software y dependiendo del tipo de análisis de código que se realice se puede obtener desde un feedback simple hasta un feedback totalmente completo del producto.

Además, el desarrollador tiene disponibles una gran multitud de herramientas que son capaces de realizar análisis de tipos diferentes calculando diferentes métricas, por lo tanto, se puede obtener un análisis muy completo haciendo uso simplemente de herramientas externas, sin ser necesario la implementación de un nuevo código que realice dichas tareas de análisis.

Finalmente, el impacto positivo que produce el uso del análisis de código en el desarrollo de un software se puede observar en la detección de errores de manera automática y fiable, generando una mejora considerable en la calidad del producto, en el rendimiento o incluso en la seguridad del mismo, abaratando considerablemente los costes.

3. Objetivos y Funcionalidades del Proyecto

3.1 Introducción

La implementación de este proyecto tiene como finalidad principal la de presentar una herramienta capaz de facilitar las tareas de corrección, a los profesores de la asignatura Proyecto de Programación de primero de Grado en Ingeniería Informática, de las prácticas que entregan los alumnos, mediante herramientas de análisis de código y la representación de los resultados en una interfaz web.

3.2 Objetivos y Funcionalidades

Como se mencionó en el capítulo 1, el objetivo principal de este proyecto es diseñar e implementar una herramienta de análisis de código para la asignatura de Proyecto de Programación de primer curso del Grado en Ingeniería Informática de la Universidad Autónoma de Madrid, que sea capaz de facilitar el trabajo de corrección de las prácticas a los profesores y que en vista al futuro pueda ser ampliable para compartir su uso con los propios alumnos.

Para alcanzar este objetivo principal, en los siguientes apartados se exponen los subobjetivos planteados, así como las funcionalidades principales que debe cubrir la aplicación que los implemente.

3.2.1 Subobjetivos

Este apartado contiene los subobjetivos que se desean alcanzar mediante. Cada una de ellos, consiste en una funcionalidad diferente, pero algunas se pueden llegar a agrupar más de una cumpliendo con la resolución del mismo subobjetivo.

- Permitir la Visualización de los Datos al Finalizar el Análisis.
- Permitir al Usuario Interactuar con los Datos Obtenidos.
- Agilizar el Proceso de Corrección del Profesor.
- Navegación Intuitiva y Clara por la Interfaz.
- Análisis Completo y Detallado de las Prácticas.

3.2.2 Funcionalidades

Para dar cabida a los anteriores objetivos, las funcionalidades que se exponen a continuación tienen como propósito principal dar una solución a las inquietudes y problemas que surgen en la corrección de las prácticas de la asignatura de Proyecto de Programación.

- **Añadir Nuevos Proyectos para el Análisis.**

- **Nuevos Proyectos.** Se permite añadir nuevos proyectos a la carpeta de proyectos para su análisis cuando esta se encuentra vacía.
- **Añadir Proyectos a Proyectos ya Existentes.** Se permite añadir nuevos proyectos a la carpeta de proyectos para su análisis cuando ya existían otros proyectos en la carpeta.

- **Sustitución de Proyectos.** Se permite eliminar los proyectos que se encontraban en la carpeta para añadir un conjunto de nuevos proyectos a analizar.
- **Selección de Proyectos a Analizar.** Se puede gestionar el contenido de la carpeta proyectos del modo que el usuario lo desee, añadiendo y eliminando de la misma los proyectos que deseen para seleccionar los proyectos a su gusto para el análisis.

- **Bienvenida e Introducción para el Usuario.** Al iniciar la interfaz web el usuario encontrará una pantalla inicial de bienvenida en la que puede encontrar una explicación introductoria de la página que va a utilizar y el motivo de su existencia
- **Elección de Apertura de Fichero de Datos.** El usuario podrá seleccionar el archivo de datos del que querrá obtener los datos a cargar para su visualización.
- **Visualización de Tablas de Datos de Proyectos.** El usuario podrá visualizar parte de los datos resultado de cada proyecto, que se obtienen del análisis, en tablas con los datos ordenados en diferentes columnas.
- **Visualización de Tablas de Datos Globales.** El usuario podrá visualizar parte de los datos globales resultado del análisis en tablas, con los datos ordenados en diferentes columnas.
- **Visualización de Gráficos de Datos de Proyectos.** El usuario podrá visualizar parte de los datos resultado de cada proyecto, que se obtienen del análisis, en gráficos circulares responsive, con los datos en una leyenda y filtrables.
- **Visualización de Gráficos de Datos Globales.** El usuario podrá visualizar parte de los datos globales resultado del análisis, en gráficos circulares responsive, con los datos en una leyenda y filtrables.
- **Visualización de Datos de Estructura de Proyectos.** El usuario podrá visualizar la estructura de los ficheros de cada proyecto directamente desde la interfaz web y de esta manera comprobar que los ficheros que contienen los proyectos son los correctos.
- **Visualización de Datos de Autores y Makefile de Proyectos.** El usuario podrá visualizar los datos correspondientes a los autores de cada proyecto directamente desde la interfaz web y también será capaz de visualizar el análisis por el que pasa el makefile de cada uno de los proyectos.
- **Obtención de Información de Contacto.** El usuario podrá obtener información de contacto que le puede ser útil mientras está utilizando la herramienta.
- **Respuestas a Preguntas Frecuentes.** El usuario tiene la opción de acceder a una sección dedicada a la resolución de cualquier cuestión que le pueda surgir de manera que pueda solucionarla instantáneamente.

4. Definición del Proyecto

4.1 Introducción

En este apartado se detalla la metodología seguida durante el transcurso del desarrollo del proyecto, explicando en qué consiste cada una de las etapas y la manera en la que se han abordado las mismas. Además, también se detalla la planificación que se ha seguido durante el desarrollo del proyecto indicando el tiempo dedicado a cada una de las tareas y mencionando las diferentes reuniones y revisiones que han tenido lugar entre ellas y los temas que se han tratado en las mismas.

4.2 Metodología

La metodología de desarrollo de software utilizada para la realización de este proyecto sigue el ***Modelo de Cascada***, que consiste en que cada una de las etapas del proyecto se realiza tan pronto como la anterior haya terminado, de manera incremental y ordenada. En el caso de este proyecto, el Modelo de Cascada utiliza la realimentación, de manera que cuando se detecta algún imprevisto o se debe realizar algún cambio, se pueda volver a cualquiera de las etapas anteriores para incluir dicho cambio en las etapas anteriores.

A continuación, en la ***Figura 4.1*** se puede observar un pequeño diagrama que explica de manera visual el ***Modelo de Cascada con Realimentación***. [15] [16] [18] [40] [41]

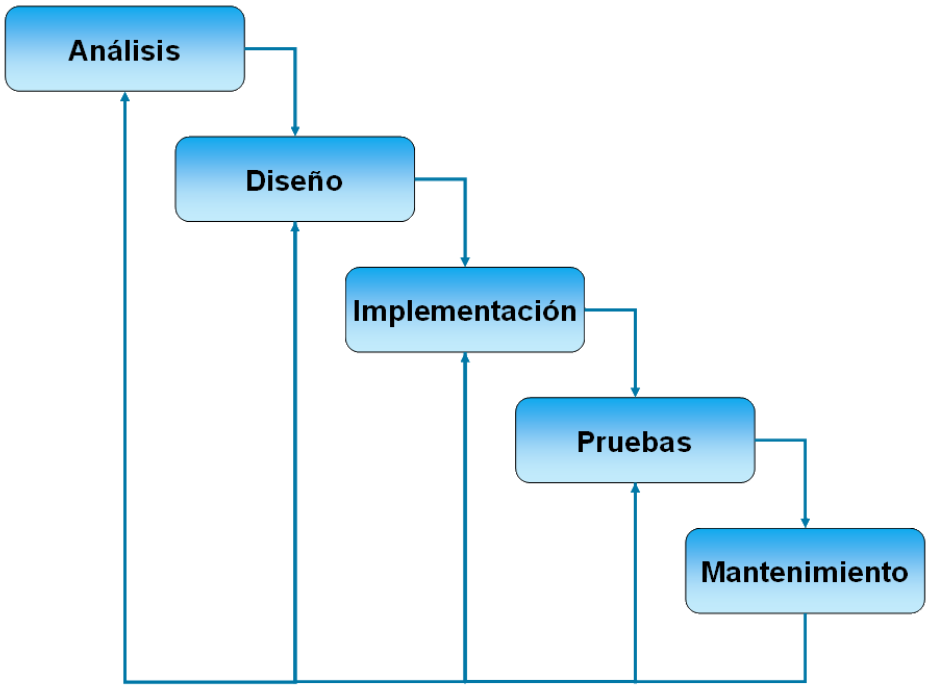


Figura 4.1: *Modelo de Cascada con Realimentación*

La distribución del trabajo realizado en cada una de las etapas que se siguen durante el desarrollo del proyecto ***CheckOode*** es el siguiente:

1. **Análisis.**

   En esta etapa del proyecto se realiza un análisis completo de los requisitos, tanto funcionales como no funcionales, del proyecto, siendo extremadamente necesario y delicado, pues es en esta etapa en la que se estimarán de manera rigurosa las necesidades del proyecto y se podrá obtener también una estimación del tiempo y del coste del desarrollo del mismo.

2. **Diseño.**

   En esta etapa de diseño se describe la estructura interna del proyecto, así como las diferentes entidades que lo componen. Es en esta etapa en la que se deciden las herramientas que se van a utilizar para dar una solución a los requisitos obtenidos en la fase anterior, la de análisis. Además, es en este momento cuando se realiza el diseño de los diferentes diagramas del proyecto, así como de las maquetas de la interfaz que se va a utilizar para mostrar los resultados con los datos devueltos por ***CheckOode***.

3. **Implementación.**

   La etapa de implementación es la etapa en la que el proyecto comienza a tomar verdadera forma y comienza a tener funcionalidad, puesto que es en esta etapa donde, haciendo uso de las etapas anteriores de captación de requisitos y diseño de las estructuras para la resolución de los mismos, empieza la programación de dichos requisitos.

   Es importante mencionar que esta es posiblemente la etapa más extensa del proyecto, puesto que en ella debe realizarse toda la programación mencionada anteriormente y todo aquello que se haya programado debe tener funcionalidad, y además, dicha funcionalidad debe ser exactamente la especificada en las etapas anteriores. Es por ello que en esta etapa, también debido a que la metodología sigue el Modelo de Cascada con Realimentación, se deben realizar pruebas constantes para verificar que la implementación va cumpliendo con los objetivos y puede seguir progresando.

4. **Pruebas.**

   En esta etapa, aunque ya en la anterior han tenido lugar algunas de ellas, se realizan todas las pruebas que debe cumplir la funcionalidad de la herramienta. Dichas pruebas se realizan directamente haciendo uso de la funcionalidad que se ha implementado y comprobando que no se obtienen situaciones de error, que la funcionalidad es la especificada en las anteriores etapas y que los resultados que se obtienen son los esperados.

   En caso de que en algún momento el resultado de alguna de las pruebas que se realizan no sea el esperado, será necesario volver a la etapa de implementación para corregir el error. Es por esto que el Modelo de Cascada con Realimentación es importante para que el producto final sea lo más satisfactorio posible.

5. **Documentación.**

   En el caso de este proyecto, la etapa de mantenimiento del Modelo de Cascada con Realimentación no llega a abordarse, debido a que la realización del proyecto está pensada hasta el momento de entrega del Trabajo de Fin de Grado. Dicha etapa sería abordable una vez se decida comenzar a utilizar la herramienta ***CheckOode*** de manera real, y debería ser abordada por el equipo docente de la propia asignatura. Entre tanto, ya que la etapa de mantenimiento no ha sido abordada, ha sido sustituida por una etapa de documentación, que consiste en la realización de este mismo documento, que engloba todo lo relativo al desarrollo del proyecto, su funcionalidad, sus objetivos, su diferentes etapas e incluso un punto dedicado exclusivamente a las líneas futuras del proyecto donde se describe también el mantenimiento que se le debe dar al proyecto.

4.3 Planificación

4.3.1 Reuniones Y Revisiones

Ya que el proyecto ha sido realizado únicamente por un individuo, no se han realizado reuniones de equipo ni ha sido necesario organizar revisiones para unificar el proyecto o realizar tareas grupales. Por el contrario, se han realizado diferentes reuniones con el tutor, que son las que se muestran a continuación.

- **Octubre 2017.**

  Tiene lugar una reunión que engloba el primer contacto con el tutor del Trabajo de Fin de Grado. Éste, expone el tema del trabajo y la forma en la que debería desarrollarse, explicando también la metodología, herramientas, lenguajes y resultado final que deberían seguirse y obtenerse.

- **Enero - Febrero 2018.**

  Tiene lugar una revisión que tiene como objetivo principal ofrecer al tutor un prototipo inicial, después de haber realizado las fases iniciales del proyecto y haber comenzado con la implementación de una primera versión. En este punto, se ponen en común las diferentes ideas, se realiza una evaluación del trabajo realizado hasta el momento y finalmente, se especifican las modificaciones y ampliaciones que deben realizarse, así como la planificación que se debe seguir a partir de ese momento.

- **Mayo - Junio 2018.**

  Tiene lugar una revisión del producto desarrollado, en su versión previa a la versión final, y de este mismo documento con el objetivo de poner al tanto, al tutor, sobre el trabajo realizado y obtener un feedback por su parte. Se concretan ciertas modificaciones a realizar y algunas ampliaciones. Además se detallan ciertos puntos a tener en cuenta en la redacción del documento y se plantea una posible reunión extra para verificar que la versión final está completa y cumple con las especificaciones.

4.3.2 Planificación y Diagrama Gantt

Los Diagramas de Gantt permiten realizar un seguimiento gráfico del desarrollo de un proyecto mediante la visualización de las tareas del mismo y las dependencias temporales y recursos que estas necesitan.

Por lo tanto, un Diagrama de Gantt ofrece las siguientes utilidades:

- Facilita la planificación
- Calcula costes, impacto de cambio, etc. más rápidamente
- Detecta inconsistencias y problemas
- Monitoriza el progreso
- Comunica la planificación a todo integrante del proyecto

A continuación se detalla el Diagrama de Gantt inicial con la planificación seguida en el proceso de desarrollo del proyecto. Como en todo proyecto, algunas temporalidades establecidas inicialmente en la planificación han cambiado (aunque solo en cuestión de días) para el desarrollo de alguna de las fases, pero el diagrama muestra la planificación establecida mensualmente y las modificaciones o alteraciones no han sido lo suficientemente significativas como para alterarlo.

Como puede apreciarse, no se trata de un diagrama orientativo de la planificación, sino que en él se muestran las fases más importantes del proyecto (análisis, diseño e implementación). [28] [36] [42]

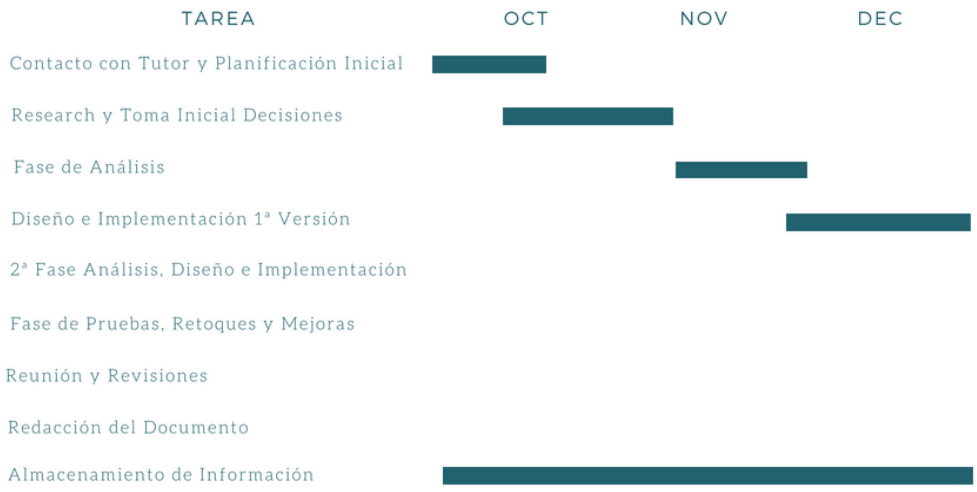


Figura 4.2: *Diagrama de Gantt parte 1*

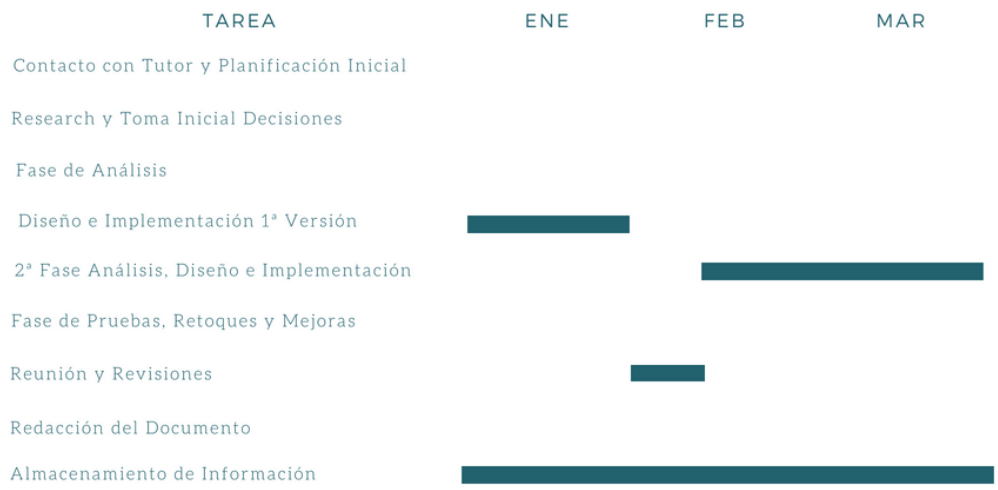


Figura 4.3: *Diagrama de Gantt parte 2*

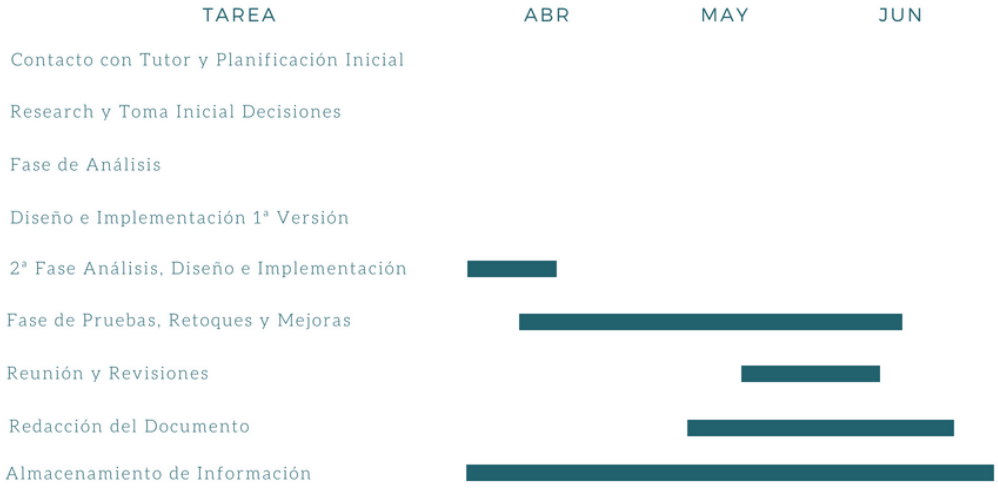


Figura 4.4: *Diagrama de Gantt parte 3*

4.4 Herramientas Utilizadas

4.4.1 Plataformas

Bitbucket

Servicio web gratuito de control de versiones que ofrece repositorios privados y permite la colaboración de diferentes usuarios en un mismo repositorio. Basado en sistemas de control de versiones Mercurial y Git. [10]

Mendeley

Aplicación web y de escritorio que sirve para el almacenamiento de referencias y documentos en base a búsquedas realizadas. Además, se permite la colaboración de varios usuarios para poder compartirlas.

Google Drive

Servicio gratuito de almacenamiento de archivos con control de versiones que dispone, además, de aplicaciones que permiten editar documentos, presentaciones y hojas de cálculo. [37]

4.4.2 Frameworks

W3Schools

Sitio web que ofrece a los usuarios tutoriales HTML, CSS, JavaScript, SQL, PHP y otras tecnologías y que presenta ejemplos de código que los usuarios pueden editar y ejecutar en un entorno de pruebas. [9] [12] [38]

SublimeText

Editor de texto y código fuente que dispone de diferentes plugins. Inicialmente desarrollado como una extensión de Vim y que con el tiempo ha ido creando identidad propia. Se puede descargar e instalar de forma gratuita pero precisa de una licencia para su uso continuado. [12] [39]

Cacoo

Software web utilizado para diseño y creación de diagramas. Posee diferentes posibilidades como flowcharts, diagramas UML, tablas y gráficos, sitemaps, y maquetación de wireframes. Permite la colaboración en tiempo real para varios usuarios. [2] [12]

Google Charts de Google Developers

Aplicación de Google que permite realizar estadísticas web y que se usa en muchos campos, como por ejemplo Google Analytics. Acepta diferentes formatos, como JSON, JavaScript y diferentes plugins integrables. Proporciona una manera de visualizar los datos en aplicaciones web mediante el uso de diferentes tipos de tablas y gráficas. [35]

4.4.3 Lenguajes de Programación

Bash Script

Programa informático que interpreta órdenes y lenguaje de consola. Es una Shell de Unix compatible con el intérprete de comandos por defecto en la mayoría de las distribuciones Linux. La especificación de la sintaxis de órdenes Bash se puede encontrar en el Bash Reference Manual distribuido por el proyecto GNU. [8]

CSS

Hojas de estilo en cascada. El CSS es un lenguaje de diseño gráfico que se encarga de definir y estructurar un documento escrito en lenguaje marcado, como HTML. Se suele utilizar para establecer el diseño visual de documentos web e interfaces de usuario. También es aplicable a documentos XML. [6] [7]

HTML

Lenguaje de programación utilizado para la implementación de páginas web. Es el encargado de dar una estructura e introducir el contenido de dichas páginas mediante el uso de las estructuras definidas que maneja. Se suele combinar con otros lenguajes como CSS, PHP y JavaScript. [5]

JavaScript

Lenguaje de programación interpretado. Se define como un lenguaje orientado a objetos que permite mejoras en la interfaz de usuario y páginas web. Se diseñó con sintaxis similar a C pero adopta convenciones del lenguaje Java, sin embargo Java y JavaScript tienen semánticas y propósitos diferentes. [34]

4.4.4 Herramientas de Análisis de Código

Pmccabe

Herramienta de análisis de código que analiza ficheros de programación escritos en lenguaje C y C++, comúnmente, y que devuelve estadísticas como por ejemplo: [4]

- Número de líneas de una función
- Línea de comienzo de las funciones
- Número de declaraciones de una función
- Complejidad Ciclomática Tradicional de una función
- Complejidad Ciclomática Mcabe de una función

Ctags

Herramienta de indexación de código, la cual analiza ficheros de código y extrae, según los parámetros con los que se invoque, aquellos elementos del código como funciones, variables, estructuras, clases, etc. Gracias a él, en éste TFG se usará como herramienta para obtener un conjunto de estadísticas de los mismos o un conjunto de fragmentos de código que siguen unos patrones determinados, como por ejemplo: [3]

- Identificación de Números Mágicos
- Uso de recursos
- Declaraciones de funciones en el código
- Detección de estructuras

4.4.5 Herramientas

Terminal Linux

La terminal es la consola del sistema interna al kernel de Linux. La consola de Linux proporciona una forma para que el kernel y otros procesos envíen mensajes de texto al usuario y para que el usuario pueda recibir mensajes de texto. El usuario generalmente ingresa texto con un teclado de computadora, los comandos, y lee el texto de salida en un monitor de computadora. [11] [13]

5. Análisis

5.1 Introducción

En este apartado se explica la etapa de análisis del proyecto, en la que principalmente se especifican los requisitos del proyecto, tanto funcionales como no funcionales. Además, en este apartado se muestran también las acciones que pueden llevar a cabo los usuarios y que, al igual que los requisitos, han sido definidas en esta fase de análisis del proyecto, una de las más importantes debido a que las decisiones tomadas son cruciales para llevar a cabo un correcto desarrollo del proyecto en el resto de la etapas.

5.2 Roles de Usuario

Los usuarios que van a hacer uso del proyecto final son en principio únicamente profesores, por lo tanto en este apartado sólo se encontrará un rol de usuario:

- Profesor

Es posible que en el futuro se quiera dar acceso también a los alumnos, en tal caso se podrían encontrar dos roles de usuario:

- Profesor
- Alumno

Es importante mencionar, que en caso de que se contemple la opción mencionada anteriormente, el diagrama de casos de uso para el alumno sería exactamente el mismo que el del profesor a menos que en líneas futuras se decida realizar modificaciones que no permitan al usuario, en caso de ser alumno, hacer uso de todas los requisitos del proyecto.

5.3 Diagramas Casos de Uso

En este apartado se explican detalladamente los casos de uso que existen en el proyecto.

- **Casos de Uso del Profesor**

  En el diagrama de la ***Figura 5.1*** se pueden observar los casos de uso que puede llevar a cabo un profesor.

  Para que un profesor sea capaz de utilizar correctamente la funcionalidad definida para este proyecto es necesario que siga unas directrices para realizar una correcta práctica al utilizar la herramienta ***CheckOode***.

  Aunque un profesor puede acceder directamente a la interfaz web del proyecto, cualquiera de los ficheros HTML, para poder utilizarla correctamente y visualizar un análisis real de los proyectos, debe haber ejecutado anteriormente el

script encargado del análisis de código y/o, en caso de tener previamente un fichero JSON con los resultados de otro análisis, deberá abrir la interfaz web desde el fichero index.html, para que esta se inicie en su página principal y posteriormente permita al usuario acceder a todo el resto de sus diferentes pantallas y utilizar sus funcionalidad, como por ejemplo, navegar hasta la pantalla del fichero estr_proy.html, abrir un fichero JSON con los datos de un análisis y poder visualizar los resultados del mismo en dicha pantalla de la interfaz.

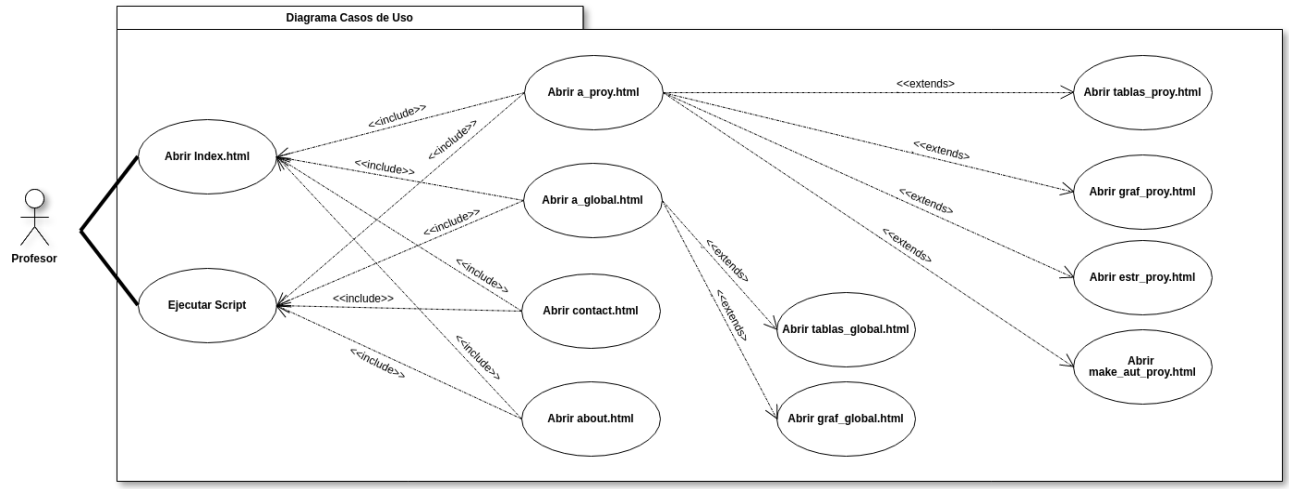


Figura 5.1: *Diagrama de Casos de Uso para el Rol de Profesor*

5.4 Catálogo y Definición de Requisitos

En este apartado del documento se exponen los requisitos funcionales y no funcionales del proyecto. Los requisitos funcionales son los que especifican las funcionalidades que debe tener el proyecto y los requisitos no funcionales son los objetivos que se deben cumplir pero que no tienen ninguna funcionalidad. [40] [41]

5.4.1 Requisitos Funcionales

- **RF.1. Inserción de Proyectos para el Análisis.** El usuario debe ser capaz de poder añadir proyectos a la carpeta de proyectos de la que se obtienen los proyectos a analizar.
- **RF.2. Ejecución del Script de Análisis.** El usuario debe ser capaz de ejecutar el script de análisis de datos desde una terminal del Sistema Operativo Linux situándose en el directorio correspondiente.
    - **RF.2.1 Ejecución de Herramienta Pmccabe.** El script debe ser capaz de utilizar correctamente la herramienta Pmccabe. Para ello, es necesario que la herramienta esté instalada.
    - **RF.2.2 Ejecución de Herramienta Ctags.** El script debe ser capaz de utilizar correctamente la herramienta Ctags. Para ello, es necesario que la herramienta esté instalada.
    - **RF.2.3 Ejecución Comandos Manipulación de Datos.** El script debe ser capaz de realizar la ejecución de los comandos que utiliza para realizar el análisis de código sin devolver errores.

- **RF.3. Acceso a la Interfaz de Usuario.** El usuario debe ser capaz de abrir la interfaz de usuario mediante la apertura del fichero correspondiente a la pantalla principal de la misma y se debe abrir automáticamente al finalizar la ejecución del script.
- **RF.4. Navegación Intuitiva por la Interfaz de Usuario.** La interfaz de usuario debe permitir al mismo una navegación clara e intuitiva por todas las pantallas de la misma, sin negar el acceso a ninguna de ellas en ningún caso.
- **RF.5. Carga del Fichero de Datos.** La interfaz debe permitir al usuario cargar el fichero JSON que desee para poder visualizar los datos en las diferentes pantallas de la interfaz.
- **RF.6. Acceso a Páginas Relacionadas.** El usuario debe ser capaz de acceder a las páginas relacionadas mediante los enlaces que se encuentran en la interfaz de usuario clicando sobre estos. En la mayoría de casos, estos enlaces aparecen en forma de imágenes.
- **RF.7. Acceso y Visualización Estructura de Proyectos del Análisis de Proyectos.** El usuario debe ser capaz de visualizar los resultados, del análisis, correspondientes a la estructura de los proyectos, en donde se deberá ofrecer una lista de los ficheros que forman el proyecto.
- **RF.8. Mostrar/Ocultar Proyectos Analizados en Análisis de Estructura de Proyectos.** El usuario debe ser capaz de poder mostrar u ocultar los diferentes proyectos que hayan sido analizados para poder visualizar cada uno de ellos de manera individual y no sobrecargar la pantalla.
- **RF.9. Acceso y Visualización de Tablas del Análisis de Proyectos.** El usuario debe ser capaz de visualizar los resultados del análisis, mostrados en tablas, de los proyectos. Se deberán mostrar en estas tablas datos como el número de línea de declaración de funciones, número de líneas de cada función, números mágicos detectados en los ficheros, recursos detectados en los ficheros, etc.
- **RF.10. Mostrar/Ocultar Proyectos Analizados y Tablas del Análisis de Proyectos.** El usuario debe ser capaz de poder mostrar u ocultar los diferentes proyectos que hayan sido analizados para poder visualizar cada uno de ellos de manera individual y no sobrecargar la pantalla.
- **RF.11. Acceso y Visualización de Gráficos del Análisis de Proyectos.** El usuario debe ser capaz de visualizar los resultados del análisis, mostrados en gráficos, de los proyectos. Se deberán mostrar en estos gráficos datos sobre la complejidad ciclomática, número de líneas de declaración de las funciones, número de líneas de las funciones, etc.
- **RF.12. Mostrar/Ocultar Proyectos Analizados y Gráficos del Análisis de Proyectos.** El usuario debe ser capaz de poder mostrar u ocultar los diferentes proyectos que hayan sido analizados para poder visualizar cada uno de ellos de manera individual y no sobrecargar la pantalla.
- **RF.13. Acceso y Visualización de Análisis Makefile y Autores.** El usuario debe ser capaz de visualizar los resultados del análisis correspondientes al makefile y los autores de los proyectos. Se deberán mostrar los nombres de los autores de cada proyecto además de las advertencias sobre las reglas de los makefiles.
- **RF.14. Mostrar/Ocultar Proyectos Analizados en el Análisis de Makefile y Autores.** El usuario debe ser capaz de poder mostrar u ocultar los diferentes proyectos que hayan sido analizados para poder visualizar cada uno de ellos de manera individual y no sobrecargar la pantalla.

- **RF.15. Acceso y Visualización de Tablas del Análisis Global.** El usuario debe ser capaz de visualizar los resultados del análisis global mostrados en tablas.
- **RF.16. Acceso y Visualización de Gráficos del Análisis Global.** El usuario debe ser capaz de visualizar los resultados del análisis global mostrados en gráficos.
- **RF.17. Interactuar con Tablas de Análisis de Proyectos.** El usuario aparte de poder visualizar los gráficos obtenidos en el análisis de proyectos debe ser capaz de interactuar con ellos, de manera que pueda ordenar los resultados y seleccionarlos.
- **RF.18. Interactuar con Tablas de Análisis Global.** El usuario aparte de poder visualizar los gráficos obtenidos en el análisis global debe ser capaz de interactuar con ellos, de manera que pueda ordenar los resultados y seleccionarlos.
- **RF.19. Interactuar con Gráficos de Análisis Proyectos.** El usuario aparte de poder visualizar los gráficos obtenidos en el análisis de proyectos debe ser capaz de interactuar con ellos, de manera que pueda filtrar los resultados y seleccionarlos.
- **RF.20. Interactuar con Gráficos de Análisis Global.** El usuario aparte de poder visualizar los gráficos obtenidos en el análisis global debe ser capaz de interactuar con ellos, de manera que pueda filtrar los resultados y seleccionarlos.

5.4.2 Requisitos No Funcionales

- **RNF.1. Apariencia Amigable e Intuitiva para el Usuario.**
- **RNF.2. Descripciones Claras de las Pantallas de la Interfaz de Usuario.**
- **RNF.3. Información de Interés para el Usuario.**
- **RNF.4. Presentación de Ayuda para el Usuario.**
- **RNF.5. Compatibilidad con Diferentes Navegadores.**

6. Diseño

6.1 Introducción

En este apartado del proyecto se explica con detalle la etapa de diseño del proyecto y se incluyen en el mismo todas las especificaciones relativas a la arquitectura de diseño del mismo junto con una galería de las diferentes maquetas pantallas de la interfaz web de ***CheckOode*** explicadas con detalle.

Además, también se puede encontrar en este apartado un subapartado que explica la arquitectura de la interfaz de usuario que incluye un diagrama de tipo sitemap de la misma.

6.2 Arquitectura

La arquitectura que se ha diseñado para este proyecto sigue un patrón modular, basado en el patrón de arquitectura de software ***Modelo, Vista, Controlador.*** Esto permite que los distintos componentes de la arquitectura del proyecto sean independientes permitiendo su adaptación, mediante pequeños cambios, a un nuevo uso.

En bases generales, el patrón ***Modelo, Vista, Controlador*** está compuesto de un modelo, que es el encargado de trabajar con los datos, de una vista, que es la encargada de representar las salidas en una interfaz, y finalmente, un controlador, que es el encargado de realizar la unión entre el modelo y la vista. De esta manera tenemos una división del código del proyecto en tres capas diferentes, cada una encargada de realizar su propia funcionalidad. [14]

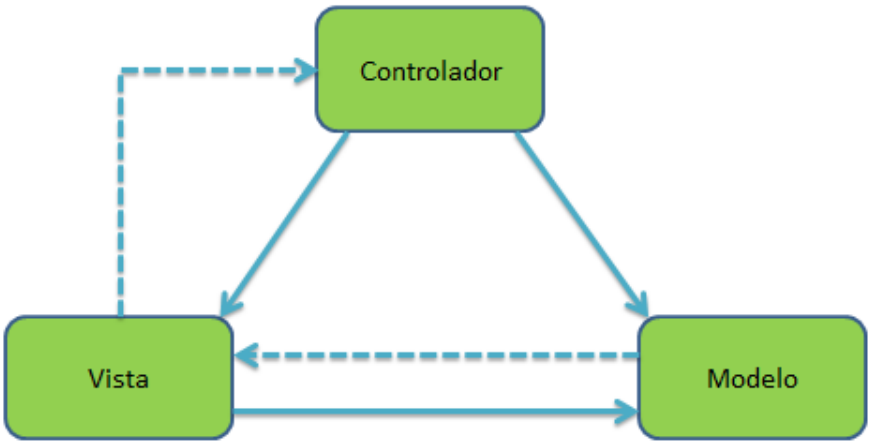


Figura 6.1: *Modelo, Vista, Controlador*

6.2.1 Modelo

Se ha establecido que el modelo del proyecto sea el módulo que contenga todos los datos relativos al análisis de código realizado. Estos datos son los que se obtienen tras la manipulación de las herramientas de análisis de código que maneja el controlador y que se almacenan estructuradamente en diferentes archivos JSON, cada uno correspondiente a uno de los análisis que se hayan realizado.

El modelo, como se menciona anteriormente, se controla mediante el controlador, que es el encargado de construirlo siguiendo un patrón que depende del tipo de análisis que se haya querido realizar y de cómo haya sido la manipulación de los datos que se analizan.

En el caso de ***CheckOode***, el modelo está muy relacionado con el controlador y con la vista debido a que el tratamiento de los datos y su representación tienen que ser muy concretos y deben manipularse los datos y estructurarse de una manera muy precisa. Es por esta razón, que aunque el proyecto sigue un patrón modular, debido a que se puede reutilizar para otras funcionalidades haciendo pequeños cambios, tanto el modelo, como la vista y como el controlador están muy unidos.

6.2.2 Vista

El módulo correspondiente a la vista se ha diseñado para que sea una representación visual, mediante una interfaz web, que muestre todos los datos que son resultado del análisis que se ha realizado.

La vista del proyecto se basa en la lectura de los datos desde un archivo JSON, que debe tener una estructura concreta, y en la posterior representación de los mismos mediante entradas de texto, en las que se exponen los datos del análisis cuyo tipo es una cadena de caracteres, mediante tablas para otros datos de texto, creadas con la herramienta de *Google Charts*, y mediante gráficos para datos numéricos, creados también mediante la herramienta de *Google Charts*.

Además, la vista de este proyecto debe cumplir con los requisitos que se han detallado anteriormente, debe tener una apariencia amigable para el usuario y ofrecer también diferentes funcionalidades que no tienen relación con el análisis de datos pero que pueden ser de interés para el usuario.

Finalmente, mencionar que esta interfaz es modificable para mostrar los datos contenidos en ficheros JSON con la misma estructura aunque hayan sido creados u obtenidos de una manera diferente a como se obtienen en este proyecto y que además se pueden realizar pequeños cambios para cambiar las funcionalidades que no están relacionadas con el análisis de datos por otras necesarias para otros proyectos.

Es de suma importancia mencionar que la vista de la herramienta basa su estructura en una plantilla de código HTML y CSS que se ha obtenido como código libre de la página W3Schools, y que posteriormente ha sido modificada y ampliada en base a las necesidades que requiere la interfaz de usuario y que han sido definidas en los requisitos para su implementación por parte del desarrollador.

6.2.3 Controlador

La funcionalidad principal del análisis de código reside en el controlador, pues es el encargado de ejecutar todas las herramientas que se utilizan para el análisis de código y posteriormente, mediante la manipulación de los datos que se obtienen de la ejecución de dichas herramientas, es el encargado de crear el modelo del proyecto, que como se ha explicado anteriormente, consiste en archivos de tipo JSON estructurados de una manera específica para que puedan ser gestionados posteriormente por la vista.

El controlador está compuesto por un Bash Script que utiliza herramientas dedicadas específicamente al análisis de código, como son Pmccabe y Ctags, utiliza comandos Linux de tratamiento de ficheros y se encarga de componer los archivos JSON con los resultados de los análisis.

El controlador del proyecto debe utilizarse desde la terminal de Linux ejecutando el Bash script correspondiente al mismo.

6.3 Estructura y Diseño de la Interfaz de Usuario

El la ***Figura 6.2*** se puede observar un sitemap completo de la estructura de la interfaz de usuario en el que se muestran las uniones para la navegación que existen entre las diferentes pantallas de la misma.

Además, se explica de forma detallada el diseño que se ha seguido para la interfaz de usuario, indicando cuáles son los componentes que están a disposición del usuario y cuál es su funcionalidad.

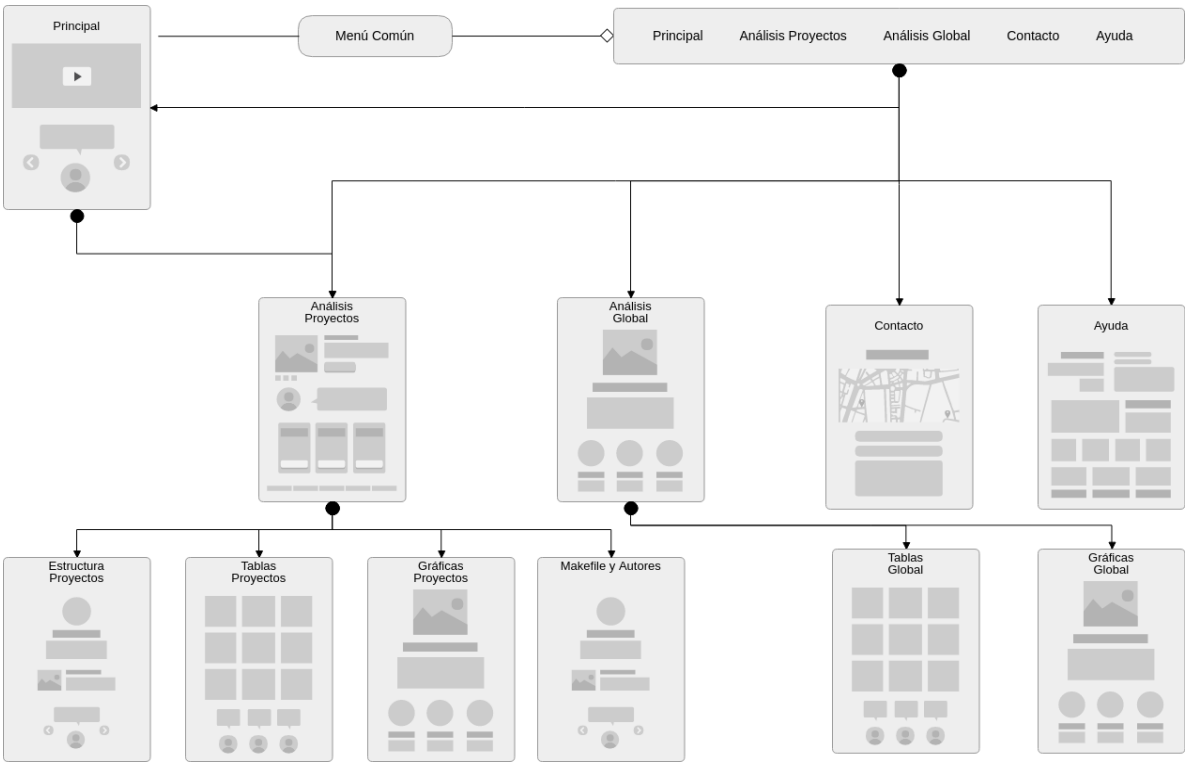


Figura 6.2: *Sitemap Interfaz Usuario*

➔ **Principal.**

Una vez abierta la interfaz de usuario, este mismo debería encontrarse en la pantalla principal de la interfaz. Dicha pantalla tiene como funcionalidad dar una bienvenida al usuario y darle también una pequeña introducción exponiendo una explicación de lo que hace la interfaz que está apunto de utilizar, de qué está compuesta, qué herramientas utiliza y de cómo debe ser utilizada ésta misma. Además, dicha pantalla da acceso, mediante imágenes clickables, a algunas páginas web que pueden ser de interés para el usuario, como por ejemplo la web de la Universidad Autónoma de Madrid o la web de la propia Escuela Politécnica Superior de la misma. Finalmente, en esta pantalla al igual que en todas las demás se dispone también de un menú horizontal desplegable que permite al usuario acceder a la pantalla de la interfaz que desee.

➔ **Análisis Proyectos.**

Si el usuario se encuentra en esta pantalla de la interfaz podrá observar una descripción que le explicará en qué consiste esta sección de análisis de proyectos. Por otra parte, el usuario tendrá la posibilidad de acceder a los cuatro diferentes tipos de análisis de proyectos existentes mediante las imágenes clickables que se encuentran en la parte inferior de la pantalla. Si lo desea, el usuario podrá acceder a cualquiera de estos tipos de análisis de proyectos o a cualquier otra pantalla de la interfaz accediendo mediante el menú horizontal desplegable que se encuentra en todas las pantallas de la interfaz.

◆ **Estructura Proyectos.**

Cuando el usuario navegue hasta esta sección del análisis de proyectos se encontrará inicialmente con un botón que le pedirá seleccionar un archivo para mostrar los resultados. Una vez el usuario pulse este botón se abrirá una pestaña que le permitirá seleccionar el archivo JSON que desee abrir. Una vez seleccionado, el usuario volverá a la pantalla anterior pero el botón habrá desaparecido y en su lugar se encontrarán los diferentes proyectos que se han analizado. Pulsando en cualquiera de estos proyectos se desplegarán y mostrarán la estructura de ficheros que los componen.

Es importante mencionar que desde esta pantalla de la interfaz y en cualquier momento, el usuario puede navegar a donde quiera mediante el menú horizontal desplegable que se encuentra en la parte superior de la pantalla.

- **Tablas Proyectos.**

  Otra de las pantallas en las que se puede encontrar el usuario es la pantalla de tablas de proyectos, que al igual que la de estructura de proyectos presenta un botón que al pulsarlo permite escoger el fichero JSON que se desea abrir para poder observar los resultados. Una vez seleccionado dicho fichero, se volverá a la pantalla anterior y se observará el mismo comportamiento que en la pantalla de estructura de proyectos, desaparecerá el botón y aparecerá un texto clickable que al pulsar desplegará el nombre de todos los proyectos que se han analizado. Posteriormente, si el usuario pulsa en cualquiera de ellos se abrirán los nombres de los ficheros del proyecto pulsado y pulsando en cualquiera de estos ficheros se abrirán las tablas con los resultados correspondientes a dicho fichero, que son resultado del uso de la herramienta *Google Charts*.

  Al igual que se ha mencionado anteriormente, desde esta pantalla de la interfaz se puede acceder a cualquier otra mediante el menú horizontal desplegable que se encuentra en la parte superior de la pantalla.

- **Gráficos Proyectos.**

  La pantalla correspondiente a la representación de los gráficos de los proyectos sigue la misma lógica que la de tablas de proyectos, pero en esta ocasión, en lugar de obtener tablas con resultados, lo que se obtiene son gráficos circulares con los datos resultado del análisis. Dichos gráficos son interactivos y se obtienen utilizando la herramienta *Google Charts.*

  Recordando también lo mencionado en los anteriores apartados es posible acceder a cualquier otra pantalla de la interfaz en cualquier momento mediante el menú horizontal desplegable que se encuentra en la parte superior de la pantalla.

- **Makefile y Autores.**

  Finalmente, la última pantalla que se explica, que pertenece al bloque del análisis de proyectos, es la de Makefile y autores. En esta pantalla se observará el botón que permite cargar el archivo JSON con los resultados y una vez pulsado y cargado dicho archivo, al igual que la lógica que siguen el resto de pantallas, el botón desaparecerá y se mostrarán en su lugar los nombres de los proyectos analizados. Una vez el usuario pulsa cualquiera de ellos se desplegará una pequeña sección que contendrá el nombre de los autores del proyecto y las observaciones correspondientes al Makefile del mismo.

  Adicionalmente, se recuerda que desde este punto, cualquier otra pantalla es accesible mediante el menú horizontal desplegable que se encuentra en la parte superior de la pantalla.

➔ **Análisis Global.**

Una vez el usuario accede a esta pantalla de la interfaz encontrará una descripción de la funcionalidad que ofrece esta sección y en el margen derecho de la pantalla un enlace a las pantallas que están relacionadas con la misma. Al pulsar en estos enlaces se puede acceder a las otras dos pantallas de la interfaz que son las encargadas de mostrar los resultados del análisis global. Dichas pantallas son la de tablas global y la de gráficos global.

Además, si el usuario lo desea podrá acceder a cualquiera de estos tipos de análisis globales o a cualquier otra pantalla de la interfaz accediendo mediante el menú horizontal desplegable que se encuentra en todas las pantallas de la interfaz.

◆ **Tablas Global.**

En esta sección del análisis global se sigue la misma lógica que en la sección de tablas del análisis de proyectos. Inicialmente, se muestra la pantalla en la que se pide añadir un archivo JSON con los resultados del análisis. Posteriormente, una vez añadido el archivo, el botón desaparece y se muestran las tablas correspondientes al análisis global, que se crean mediante la herramienta G*oogle Charts*.

Adicionalmente, es importante recordar que desde esta pantalla se puede navegar hasta cualquier otra de la interfaz haciendo uso del menú horizontal desplegable que se encuentra en la parte superior de la pantalla.

◆ **Gráficos Global.**

Finalmente, en la última sección del análisis global se encuentra la pantalla correspondiente a los gráficos globales. En esta se muestran los gráficos creados mediante la herramienta *Google Charts* que muestran los datos que son resultado del análisis global.

También, desde esta pantalla se puede acceder a las demás mediante el menú horizontal desplegable que se encuentra en la parte superior derecha de la pantalla.

➔ **Contacto.**

La pantalla de contacto es accesible desde cualquier punto de la interfaz a través del menú horizontal desplegable que se encuentra en la parte superior de la pantalla.

Los datos de contacto que pueden ser de interés para el usuario son los que se presentan a continuación. En esta pantalla el usuario puede obtener números de teléfono, direcciones e incluso un acceso directo al correo institucional para poderse poner en contacto por ejemplo con los profesores de la asignatura vía email. Lo que es más, el usuario tiene acceso mediante imágenes clickables a las páginas web

correspondientes a la Universidad Autónoma de Madrid y a la de la Escuela Politécnica Superior de esta misma universidad, e incluso, a la sección directa de contacto de la página web de la universidad.

➔ **Ayuda.**

La siguiente pantalla, aunque no ofrece una funcionalidad específica de análisis de código, puede considerarse crucial para la interfaz, pues es la encargada de responder a cualquier duda o problema común con el que pueda toparse el usuario, permitiendo que éste pueda continuar haciendo uso de la interfaz comprendiendo su funcionalidad y haciendo un correcto uso de la misma.

Además, no contesta únicamente a preguntas relacionadas con la interfaz, sino que también contiene información relativa al proyecto en general, su motivación, las herramientas que utiliza e incluso una descripción de las mismas.

Finalmente, y como se ha mencionado en todos los puntos anteriores, es importante saber que se puede acceder, mediante el menú horizontal desplegable que se encuentra en la parte superior de todas las pantallas de la interfaz, a cualquier pantalla de la interfaz a la que el usuario desee acceder.

7. Implementación

7.1 Introducción

En este apartado se explica de una manera más detallada la fase de implementación del proyecto. Se detallan de manera individual las implementaciones correspondientes al modelo, al controlador y a la vista y se explican también de manera detallada cuáles son sus componentes, cómo se han utilizado en la implementación y cuál es la funcionalidad que tienen en la misma.

7.2 Implementación del Modelo

El modelo como tal no ha sido implementado directamente, sino que más bien, al tratarse de un archivo JSON, éste es creado, mediante la ejecución del controlador, para cada uno de los análisis que se realice. Es decir, el controlador solicitará la creación del modelo a partir de la ejecución de los scripts y herramientas externas que analizan el código. Estas extraerán la información relevante y con ello, el controlador tendrá el modelo a su disposición.

Aun así, es importante explicar que el modelo se ha diseñado para que siga un formato específico y sea capaz de almacenar los datos que han sido resultado del análisis de código para que, posteriormente, estos puedan ser evaluados y utilizados en la vista.

El modelo está estructurado de tal manera que al crearse debe contener una lista de todos los proyectos que se han analizado, con sus nombres y con los datos del análisis correspondiente a los mismos. Además, para cada uno de estos proyectos, deberá contener una subestructura con todos los ficheros que posee cada uno de ellos y dentro de cada uno de los ficheros, de nuevo, una subestructura con todos los datos del análisis correspondiente a cada uno de los ficheros.

Por lo tanto, se puede especificar, que en los ficheros JSON que son creados para cada análisis realizado, se contiene toda la información que se obtiene como resultado del análisis y que será necesaria para poder elaborar la vista y mostrar todos los resultados de manera visual al usuario.

7.3 Implementación del Controlador

El controlador del proyecto se basa en un Bash Script que se encarga de realizar todo el análisis de código de los proyectos que se deseen analizar. Su implementación se basa en el uso de lenguaje Bash Script, utilizando diferentes comandos dedicados al tratamiento de ficheros y gestionando las salidas de los mismos de tal manera que se obtengan los datos buscados en el análisis.

Además, dentro del propio script, se hace uso de las dos herramientas de análisis de código que se utilizan en el proyecto, Pmccabe y Ctags. Estas herramientas son ejecutadas desde el propio script y sus salidas son gestionadas en ficheros de texto, utilizando también

la herramienta AWK para el tratamiento de dichos ficheros y el posterior correcto manejo de los datos que contienen.

Los datos que son analizados por el controlador se corresponden a los nombres de los diferentes proyectos que son analizados, así como la estructura interna de los mismos, es decir, a los archivos que contienen. Además, se analizan todos y cada uno de estos archivos por separado, obteniendo los nombres de los autores de los mismos, las macros que se utilizan en cada uno de ellos, las funciones que existen en su interior, así como las líneas en las que se ha declarado cada una de ellas, el número de líneas que tienen, la complejidad ciclomática de cada función, el número de declaraciones que tienen, el uso indebido de números mágicos dentro del código e incluso los recursos que se han utilizado, como mallocs y frees.

Finalmente, el controlador también es el encargado de que a medida que se van obteniendo y gestionando los datos que son resultado del análisis de código, se vayan guardando de manera estructurada en el fichero JSON que se encarga de almacenarlos y que será utilizado por la vista para representarlos en la interfaz de usuario.

7.4 Implementación de la Vista

La implementación de la vista consiste en la implementación de una interfaz web con varias pantallas por las que el usuario debe poder navegar y que se encargaran de representar visualmente todos los datos que se han obtenido en el análisis de código realizado con anterioridad.

La implementación de la vista consiste en la implementación de diferentes archivos HTML, que utilizando hojas de estilo CSS para su diseño gráfico y utilizando también la herramienta Google Charts, sean capaces de representar los datos del análisis, ya sea de manera textual o de manera visual en tablas y gráficos de diferentes tipos.

Se han implementado varios archivos en base a una plantilla HTML y CSS, que ofrece w3schools y que se ha modificado de manera que se ajuste a la interfaz de usuario exigida para el proyecto. Cada uno de estos archivos corresponde con una de las pantallas de la interfaz de usuario y en cada uno de ellos se gestionan de manera diferente los distintos componentes que forman parte de cada pantalla. Además, la mayoría de ellos hacen uso de JavaScript para que las pantallas de la interfaz sean interactivas a las acciones del usuario.

Todas las pantallas de la interfaz se han implementado para ser accesibles desde cualquier otra pantalla, es decir, toda la interfaz es accesible desde cualquiera de las diferentes pantallas de la misma.

La vista, en sus diferentes pantallas como se ha comentado, se encarga de la representación de los datos analizados, por el controlador y se encarga de representar los datos numéricos, como el número de líneas de una función o número de línea de declaración de cada función en gráficos de tipo circular y los datos de texto en tablas de datos, con los datos ordenados en diferentes columnas.

Finalmente, cabe mencionar, que la implementación de la interfaz de usuario también se encarga de introducir todos los requisitos de ayuda a la navegación del usuario, así como los de acceso a páginas de interés relacionadas para el usuario y también el acceso a datos de contacto. Todo lo anterior ha sido implementado haciendo uso de las diferentes herramientas de lenguajes de programación que se han mencionado, como HTML, CSS, JavaScript y herramientas como Google Charts.

8. Pruebas

8.1 Introducción

En este apartado se explicará toda la información correspondiente a las pruebas que se han realizado para el proyecto, explicando el motivo por el que se han realizado y finalmente se exponen los resultados que se obtienen tras la realización de las mismas.

Es importante mencionar que las pruebas no se han realizado únicamente en la fase de pruebas del proyecto, sino que también se han ido realizando a medida que éste se iba desarrollando para verificar que los requisitos se iban cumpliendo y permitir que la implementación del proyecto pudiese avanzar.

8.2 Alcance de las Pruebas

8.2.1 Pruebas de Usabilidad

En lo correspondiente a las pruebas de usabilidad se ha comprobado que los usuarios de la interfaz sean capaces de navegar por todas las pantallas de la misma sin encontrarse con problemas y que además la navegación que realicen sea intuitiva, simple y con una correcta funcionalidad.

Además, en cuanto a la usabilidad de los usuarios, como inicialmente el proyecto está dirigido a un uso exclusivo del equipo docente, se entiende que cualquier profesor puede tener un acceso al sistema operativo GNU/Linux para poder ejecutar el script correspondiente para la realización del análisis de código.

Para comprobar que se cumplen estos criterios de usabilidad se han hecho pruebas constantes a medida que se iba desarrollando el proyecto y además al acabar, se ha presentado la interfaz de usuario a un grupo de personas para que ellos mismos, sin conocimientos previos del proyecto o de la asignatura, navegaran por las diferentes pantallas de la interfaz de usuario.

Los resultados de estas pruebas han sido satisfactorios y al finalizarlas se ha decidido que se cumplen los criterios exigidos de usabilidad.

8.2.2 Pruebas Funcionales

Las pruebas funcionales que se han realizado residen prácticamente en su totalidad en la correcta ejecución del script encargado de realizar el análisis de código y también en el funcionamiento correcto de la interfaz de usuario, que implica además que todos los datos que se obtienen del análisis de código sean representados de manera correcta y clara en dicha interfaz.

Las pruebas de funcionamiento buscan cumplir con la correcta implementación de los requisitos funcionales que se han descrito en apartados anteriores y se han llevado a cabo tanto durante el desarrollo del proyecto como en la etapa final del desarrollo del

mismo. Ha sido necesario distribuir las pruebas de esta manera para poder ir cerrando la implementación de diferentes requisitos que son necesarios para el correcto funcionamiento de otras posteriores y una vez terminado el desarrollo realizar las pruebas del conjunto de los requisitos en su totalidad.

Durante las distintas etapas del desarrollo del proyecto se han ido encontrando diferentes errores y contratiempos relacionados con el uso de las herramientas de análisis de código, con el tratamiento de los datos mediante la gestión de ficheros y también en la interfaz, sobre todo con la creación y el manejo de las tablas y los gráficos creados con google charts. Todos los errores que se han ido encontrando han sido reparados y por lo tanto, se ha conseguido alcanzar una funcionalidad correcta del proyecto, tanto de las herramientas de análisis de código como de las herramientas de representación de los resultados.

8.2.3 Compatibilidad

Las pruebas de compatibilidad más importantes que se han realizado para el proyecto residen en que la interfaz de usuario sea accesible desde diferentes navegadores, sin que la misma se vea afectada en cuanto a apariencia y funcionalidad.

Al encontrarnos realizando y utilizando el proyecto en un sistema operativo GNU/Linux, se han realizado las pruebas de compatibilidad de navegadores con navegadores Google Chrome y Mozilla Firefox, dando como resultado una compatibilidad satisfactoria.

8.3 Estrategia y Desarrollo

8.3.1 Estrategia

La estrategia que se ha seguido para la realización de las pruebas depende de cuales hayan sido los requisitos que se han probado.

En el caso de la ejecución del script de análisis de código las pruebas se han realizado sobre los ficheros de salida de las diferentes herramientas de análisis de código utilizadas y también se han realizado pruebas sobre los ficheros JSON, creados por dicho script, encargados del almacenamiento de los resultados del análisis, mediante validación de dichos ficheros con un validador online de archivos de este tipo. También, en este caso, ha sido de gran ayuda la terminal de Linux, pues gracias a ella se han podido realizar pruebas directas y visuales del contenido de los diferentes ficheros que se iban gestionando y su contenido, los resultados del análisis.

Por otro lado, para la realización de las pruebas de la interfaz de usuario, se han realizado diferentes flujos de navegación por la interfaz, comprobando que en todos los puntos de la misma se cumplieran tanto el diseño como los requisitos exigidos para las mismas. Además, se han realizado también pruebas interactivas con las tablas y gráficos que contienen los resultados visuales del análisis para comprobar que su funcionamiento es el esperado y que los datos que se representan por dichas tablas y gráficas son correctos y

corresponden con los análisis de código realizados por el script y almacenados por los ficheros JSON.

8.3.2 Tipo de Pruebas

El tipo de pruebas que se han realizado se puede dividir en dos tipos de prueba específicos: las pruebas de caja blanca y las pruebas de caja negra.

En cuanto a las pruebas de caja blanca realizadas para el script, se han realizado pruebas ligadas directamente al código, es decir, ligadas a cómo es la funcionalidad del flujo del propio código programado y de las diferentes tareas que se van realizando durante su ejecución. Además en este tipo de pruebas se han realizado monitorizaciones de las entradas y salidas de datos que se iban produciendo durante la ejecución del código. En lo relacionado con las pruebas de caja blanca realizadas sobre la interfaz de usuario, se han realizado pruebas directamente relacionadas con el flujo de ejecución del código mediante la consola que ofrece el navegador de Google Chrome. También, se han monitorizado la creación y la ejecución de requisitos de las tablas y los gráficos, creados con Google Charts, mediante la visualización del código que se iba creando para su incorporación en la interfaz de usuario.

En cuanto a las pruebas de caja negra realizadas para el script, se ha comprobado que los datos que son resultado del análisis de código de un grupo de proyectos fueran los correspondientes a los datos que se habían querido analizar, de manera que según la entrada de datos para el análisis la salida fuera la esperada. En cuanto a las pruebas de caja negra realizadas para la interfaz de usuario, se ha comprobado que el diseño de la interfaz para los archivos de la misma fuera el correcto y que los datos que se visualizaban en la interfaz fueran los correspondientes a los que se cargaban al abrir el fichero de entrada de los datos resultado del análisis de código. [1]

8.4 Resultados y Conclusiones

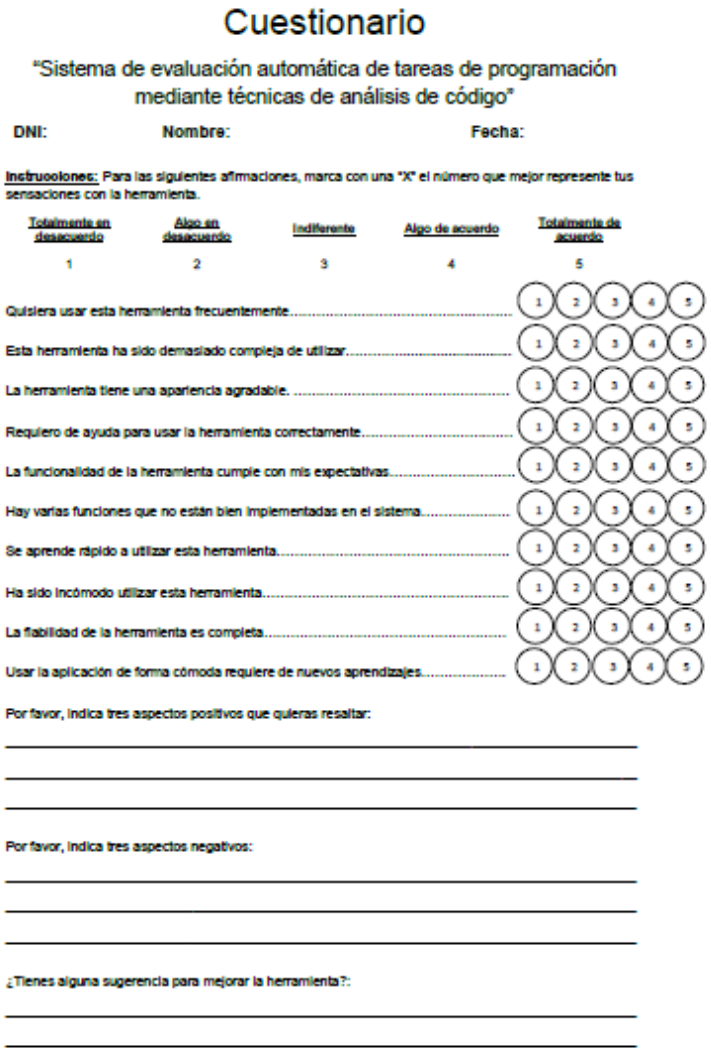
En conclusión, la fase de pruebas realizada para el proyecto ***CheckOode*** tiene como resultado una salida favorable, en la que se ha comprobado que los requisitos que debe de ser capaz de realizar el analizador se han cubierto por el mismo de manera satisfactoria, y que además la representación de todos los datos del análisis en la interfaz de usuario es correcta, incluyendo además que se cumple con todos los requisitos especificados en cuanto a su utilidad, su intuitividad y su diseño.

9. Evaluación

En las pruebas de usabilidad se ha presentado la herramienta a unos usuarios voluntarios y se ha aplicado la encuesta que permite medir la usabilidad del sistema (System Usability Scale, SUS) [43]. Para ello se ha realizado una encuesta para que estos usuarios sean los que evalúen la herramienta, puesto que solo la evaluación del desarrollador no sería igual de crítica.

La encuesta que se ha realizado a los usuarios busca obtener un feedback que permita utilizarse para realizar modificaciones y mejoras para la herramienta al igual que busca la posible detección de errores en la herramienta o dificultades de los usuarios al utilizarla.

La encuesta realizada a los usuarios consiste en medir los 10 items del SUS, más la petición de tres características a favor de la herramienta y tres características en contra y, finalmente, se pide al usuario que proponga alguna mejora para la herramienta. Dicha encuesta es la que se muestra:



Cuestionario

"Sistema de evaluación automática de tareas de programación mediante técnicas de análisis de código"

DNI: **Nombre:** **Fecha:**

Instrucciones: Para las siguientes afirmaciones, marca con una "X" el número que mejor represente tus sensaciones con la herramienta.

Totalmente en desacuerdo	Algo en desacuerdo	Indiferente	Algo de acuerdo	Totalmente de acuerdo
1	2	3	4	5

Quisiera usar esta herramienta frecuentemente..... 1 2 3 4 5

Esta herramienta ha sido demasiado compleja de utilizar..... 1 2 3 4 5

La herramienta tiene una apariencia agradable. 1 2 3 4 5

Requiero de ayuda para usar la herramienta correctamente..... 1 2 3 4 5

La funcionalidad de la herramienta cumple con mis expectativas..... 1 2 3 4 5

Hay varias funciones que no están bien implementadas en el sistema..... 1 2 3 4 5

Se aprende rápido a utilizar esta herramienta..... 1 2 3 4 5

Ha sido incómodo utilizar esta herramienta..... 1 2 3 4 5

La fiabilidad de la herramienta es completa..... 1 2 3 4 5

Usar la aplicación de forma cómoda requiere de nuevos aprendizajes..... 1 2 3 4 5

Por favor, indica tres aspectos positivos que quieras resaltar:

_____

_____

_____

Por favor, indica tres aspectos negativos:

_____

_____

_____

¿Tienes alguna sugerencia para mejorar la herramienta?:

_____

_____

Figura 9.1: *Encuesta Realizada a los Usuarios*

9.1 Resultados y Conclusiones

Como resultados a las cinco encuestas que se han realizado se han obtenido los siguientes:

Afirmaciones	Totalmente en Desacuerdo	Algo en Desacuerdo	Indiferente	Algo de Acuerdo	Totalmente de Acuerdo
Quisiera usar esta herramienta frecuentemente	**0/5**	**0/5**	**0/5**	**2/5**	**3/5**
Esta herramienta ha sido demasiado compleja de utilizar	**0/5**	**5/5**	**0/5**	**0/5**	**0/5**
La herramienta tiene una apariencia agradable	**0/5**	**0/5**	**0/5**	**3/5**	**2/5**
Requiero de ayuda para usar la herramienta	**2/5**	**3/5**	**0/5**	**0/5**	**0/5**
La funcionalidad de la herramienta cumple con mis expectativas	**0/5**	**0/5**	**0/5**	**3/5**	**2/5**
Hay varias funciones que no están bien implementadas	**4/5**	**1/5**	**0/5**	**0/5**	**0/5**
Se aprende rápido a utilizar esta herramienta	**0/5**	**0/5**	**0/5**	**1/5**	**4/5**
Ha sido incómodo utilizar esta herramienta	**2/5**	**3/5**	**0/5**	**0/5**	**0/5**
La fiabilidad de la herramienta es completa	**0/5**	**0/5**	**0/5**	**2/5**	**3/5**
Usar la aplicación de forma cómoda requiere de nuevos aprendizajes	**3/5**	**2/5**	**0/5**	**0/5**	**0/5**

Tabla 9.1: *Resultados Encuesta Realizada a los Usuarios*

10. Conclusiones y Trabajo Futuro

10.1 Conclusiones

Como conclusiones globales del proyecto que se ha desarrollado para este Trabajo Final de Grado, se expone que se han cumplido las expectativas iniciales, obteniendo una funcionalidad completa del análisis de código que se quería realizar y la capacidad de representar los resultados en una interfaz gráfica para simplificar el trabajo de corrección que deben realizar los profesores de la asignatura Proyecto de Programación.

Ha sido fundamental para el desarrollo del proyecto una gran implicación por parte del desarrollador y una búsqueda exhaustiva de información y de introducción al mundo del análisis de código para poder comprender el funcionamiento y la utilidad del mismo, conocer las ventajas que proporciona y decidir las herramientas que utilizar para la realización de este proyecto.

Como subobjetivos cumplidos en el proyecto se pueden mencionar:

- Funcionalidad completa de la interfaz de usuario.
- Análisis completo del código de los proyectos.
- Gestión y tratamiento de ficheros.
- Uso de herramientas de análisis de código.
- Uso de herramientas de representación de datos.
- Uso de diferentes lenguajes de programación.
- Simplificación de la corrección de las prácticas de la asignatura.
- Interfaz intuitiva, con un diseño amigable y con gran facilidad de manejo.
- Producto desarrollado reutilizable y ampliable.

Finalmente, el desarrollo de este proyecto ha logrado el objetivo de aprendizaje propuesto para el desarrollador, pues ha aprendido un amplio conjunto de conceptos, de métodos de análisis de código, de metodologías para el tratamiento de información y gestión de ficheros y además una amplia gama de lenguajes de programación y herramientas web para el diseño de una interfaz capaz de representar todos los datos que han sido obtenidos.

10.2 Trabajo Futuro

10.2.1 Modularidad

Una de las ventajas del desarrollo del proyecto utilizando el patrón ***Modelo, Vista, Controlador*** es la capacidad de hacer uso de los diferentes componentes del proyecto para la implementación y desarrollo de otros proyectos diferentes.

En cuanto a ***CheckOode*** se ofrece la posibilidad de hacer uso del script de análisis de código sobre otro tipo de proyectos a analizar realizando pequeñas modificaciones en la gestión y el tratamiento de los ficheros de salida del análisis de código que realiza, de

manera que los datos que se obtienen sean los correspondientes a los que se quieren analizar en dichos proyectos.

Además, la interfaz de usuario queda abierta para cualquier modificación, puesto que únicamente necesita conocer la estructura de un archivo de tipo JSON para ser capaz de representar todos los datos que éste esté almacenando mediante las mismas herramientas que utiliza actualmente y en lo que concierne a las pantallas de ayuda al usuario, que son externas a la funcionalidad, la modificación que debería realizarse es la de modificación de la información que se quiere presentar a través de la misma.

Por lo tanto, ***CheckOode*** es un proyecto que ofrece una amplia modularidad sin necesidad de realizar demasiados cambios ni de dedicar demasiado tiempo a su modificación.

10.2.2 Extensibilidad

La extensibilidad que posee el proyecto desarrollado es amplia en su totalidad, debido a que permite cualquier tiempo de modificación o ampliación de los requisitos de la misma sin la necesidad de realizar cambios en las funcionalidades que ya posee.

En cuanto a las posibles ampliaciones que se podrían realizar para ***CheckOode*** se encuentran la ampliación del análisis de código obteniendo más información correspondiente al código, y por lo tanto más resultados, y de manera conjunta, se podrían realizar ampliaciones correspondientes a la interfaz de usuario para la representación de los nuevos datos analizados o simplemente la presentación de nuevos datos de interés para los usuarios.

Es por todo esto que el proyecto que se ha desarrollado posee una amplia extensibilidad que aumenta la usabilidad y el valor de la herramienta.

10.2.3 Líneas Futuras

En relación a las líneas futuras del proyecto se plantea el mantenimiento de la herramienta de análisis de código, así como de la interfaz de usuario encargada de la representación de los datos. Además, se espera poder ampliar la funcionalidad de la herramienta de análisis para poder obtener más resultados al utilizarla y así poder simplificar, mejorar y automatizar aún más la corrección de las prácticas de la asignatura.

También, de manera continuada, se pretende mejorar el diseño de la interfaz de usuario para que la visualización de los datos sea todavía más amigable e intuitiva, además de ser capaz de representar nuevos datos del análisis y utilizar diferentes herramientas de representación de datos.

Finalmente, como una posible ampliación, cuya decisión de introducir o no deben tomar los docentes de la asignatura, se ha planteado la posibilidad de que los alumnos tengan permisos para utilizar la herramienta y puedan utilizarla durante el transcurso de la realización de sus prácticas para obtener un feedback automático del proceso de implementación de las mismas y de las diferentes versiones que vayan implementando.

BIBLIOGRAFÍA

[1] “Análisis Dinámico de Código vs Análisis Estático | Go4IT Solutions.” n.d. https://go4it.solutions/es/blog/analisis-dinamico-de-codigo-vs-analisis-estatico.

[2] “How To Make Awesome Charts For Presentations | Business Insider.” n.d. https://www.businessinsider.com.au/how-to-make-awesome-charts-for-presentations-2010-1

[3] “Hasktags: Produces Ctags “Tags” and Etags “TAGS” Files for Haskell Programs.” n.d. http://hackage.haskell.org/package/hasktags.

[4] “PMCCABE Pmccabe.1 Man Page.” n.d. https://people.debian.org/~bame/pmccabe/pmccabe.1.

[5] “¿Qué Es y Para Qué Sirve HTML? El Lenguaje Más Importante Para Crear Páginas Webs. HTML Tags (CU00704B).” n.d. http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=435:ique-es-y-para-que-sirve-html-el-lenguaje-mas-importante-para-crear-paginas-webs-html-tags-cu00704b&catid=69:tutorial-basico-programador-web-html-desde-cero&Itemid=192.

[6] W3C. 2015. “CSS Current Work & How to Participate.” *W3.Org*. http://www.w3.org/Style/CSS/current-work.en.html.

[7] w3.org. 2018. “HTML & CSS - W3C.” https://www.w3.org/standards/webdesign/htmlcss.

[8] “The A-Z of Programming Languages: BASH/Bourne-Again Shell - Computerworld.” n.d. https://www.computerworld.com.au/article/222764/a-z_programming_languages_bash_bourne-again_shell/?pp=2&fp=16&fpid=1.

[9] “Aprender a Programar a Través de Internet.” n.d. https://www.eldiario.es/turing/Aprendiendo-programar-traves-Internet_0_264223576.html.

[10] “Herramienta de Gestión de Código Git de Atlassian Bitbucket Para Equipos.” n.d. https://es.atlassian.com/software/bitbucket.

[11] “¿Qué Es La Terminal de Linux? Y Comandos Básicos Que Deberías Utilizar.” n.d. https://www.technodyan.com/que-es-la-terminal-de-linux/.

[12] Sánchez, Jordi. 2006. “¿Qué Es Un ‘Framework’? | Jordisan.Net.” http://jordisan.net/blog/2006/que-es-un-framework/.

[13] “La Terminal de Linux. Una Introducción a Una Potente Herramienta.” n.d. https://www.atareao.es/ubuntu/la-terminal-de-linux/.

[14] Álvarez, Miguel. 2014. “Qué Es MVC.” *Desarrolloweb.Com*. https://desarrolloweb.com/articulos/que-es-mvc.html.

[15] Domínguez Pablo. 2017. “En Qué Consiste El Modelo En Cascada - Gestiona Tu Proyecto de Desarrollo - OpenClassrooms.” https://openclassrooms.com/courses/gestiona-tu-proyecto-de-desarrollo/en-que-consiste-el-modelo-en-cascada.
[16] Karel Gómez. 2017. “Top 5 Metodologías de Desarrollo de Software.” https://www.megapractical.com/blog-de-arquitectura-soa-y-desarrollo-de-software/metodologias-de-desarrollo-de-software.
[17] Harman, Mark, Malcolm Munro, Lin Hu, and Xingyuan Zhang. 2002. “Source Code Analysis and Manipulation.” In *Information and Software Technology*.
[18] “Qué Metodología Será Más Adecuada Para Mi Proyecto Software.” n.d. https://es.slideshare.net/leansight/qu-metodologa-ser-ms-adecuada-para-mi-proyecto-software-13905273.
[19] Sözer, H. 2015. “Integrated Static Code Analysis and Runtime Verification.” *Software - Practice and Experience* 45 (10). John Wiley and Sons Ltd: 1359–73. doi:10.1002/spe.2287.
[20] “Static Analysis with Clang - Confessions of a Wall Street Programmer.” n.d. http://btorpey.github.io/blog/2015/04/27/static-analysis-with-clang/.
[21]Binkley, David. 2007. “Source Code Analysis: A Road Map.” *Future of Software Engineering (FOSE ’07)*. IEEE Computer Society. doi:10.1109/FOSE.2007.27.
[22]García Sánchez, Ana María. 2010. “Evaluación de Métricas de Calidad Del Software Sobre Un Programa Java.”
[23] “Bienvenido - Portada - SonarQube Hispano.” n.d. https://sonarqubehispano.org/.
[24] Arroyo, M, F Bavera, and G Regis. 2010. “Análisis Estático de Programas.” *XII Workshop de Investigadores En Ciencias de La Computación*, 1–4.
[25] “Cómo Interpretar Métricas de Calidad Software: Entendiendo El Cuadro de Mando de Sonar (1/3) - Javier Garzás.” n.d. http://www.javiergarzas.com/2013/09/metricas-sonar-1.html.
[26] “Análisis Estático de Código - Algunas Herramientas - Gustavo Terrera.” n.d. https://testingbaires.com/analisis-estatico-de-codigo-algunas-herramientas/.
[27] “Análisis Estático del Código - Raúl Expósito.” n.d. https://raulexposito.com/documentos/analisis-estatico-codigo/.
[28] “Todos Tus Diseños – Canva.” n.d. https://www.canva.com/.
[29] “6.3. La Orden Apt-Cache.” n.d. https://debian-handbook.info/browse/es-ES/stable/sect.apt-cache.html.
[30] “Simian – Web and Mobile Design & Development Studio.” n.d. https://simian.co/.
[31] Cutroni, J. 2010. “Google Analytics.”
[32] Merson, Paulo, Joseph Yoder, Eduardo Guerra, and Ademar Aguiar. n.d. “Continuous Inspection.” *Submission*.

[33] “PMD y la calidad estática del código | Marco de Desarrollo de la Junta de Andalucía” n.d. http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/374

[34] Refsnes_Data. 2012. “JavaScript Tutorial.” *W3Schools*, JavaScript Tutorial. http://www.w3schools.com/js/default.asp.

[35] Google. 2017. “Charts | Google Developers.” *Google*. https://developers.google.com/chart/.

[36] “Diagramas de Gantt Para La Gestión de Proyectos.” n.d.

[37] Google. 2016. “Google Drive: Almacenamiento En La Nube, Copias de Seguridad de Fotos, Documentos y Mucho Más.” https://www.google.com/intl/es_es/drive/.

[38] W3Schools. 2012. “W3Schools Online Web Tutorials.” *W3Schools.Com*. http://www.w3schools.com/.

[39] Sublime HQ Pty Ltd. 2017. “Sublime Text - A Sophisticated Text Editor for Code, Markup and Prose.” https://www.sublimetext.com/.

[40] Asignatura Tercer Curso - Ingeniería del Software. Grado Ingeniería Informática

[41] Asignatura Tercer Curso - Proyecto Ingeniería del Software. Grado Ingeniería Informática

[42] Asignatura Primer Curso – Proyecto de Programación. Grado Ingeniería Informática

[43] System Usability Scale n.d. https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

GLOSARIO

ACRÓNIMOS

CIL → **C**ommon **I**ntermediate **L**anguage

GUI → **G**raphical **U**ser **I**nterface

PHP → **H**ypertext **P**reprocessor

PDF → **P**ortable **D**ocument **F**ormat

CSS → **C**ascading **S**tyle **S**heets

XML → **E**xtensible **M**arkup **L**anguage

GNU → **G**NU **N**o es **U**nix

AWK → Alfred **A**ho, Peter **W**einberger y Brian **K**ernighan

ACSL→ **A**NSI/ISO **C** **S**pecification **L**anguage

JSON → **J**ava**S**cript **O**bject **N**otation

HTML → **H**yper **T**ext **M**arkup **L**anguage

DEFINICIONES

CheckOode → Es la herramienta desarrollada.

Metodología → Conjunto de métodos que se siguen en una investigación científica, un estudio o una exposición doctrinal.

Diagrama de Gantt → Herramienta para planificar y programar tareas a lo largo de un periodo de tiempo determinado.

Sitemap → Archivo en el que se reúnen todas las páginas que componen un sitio web.

Diagrama de Casos de Uso → Herramienta que contiene la descripción de los diferentes pasos o actividades que se deben realizar para llevar un proceso a cabo.

Script → Es un programa compuesto de ordines que se almacena en un archive de texto plano.

Interfaz → Conexión, física o lógica, entre una computadora y el usuario, un dispositivo periférico o un enlace de comunicaciones.

Makefile → Ficheros de texto que utiliza make para llevar la gestión de la compilación de programas.

Framework → Es una estructura conceptual y tecnológica de asistencia definida, con módulos concretos de software, que puede servir de base para la organización y desarrollo de software.

Plataforma → Entorno informático determinado, que utiliza sistemas compatibles entre sí.

Flowchart → Es un tipo de diagrama que representa un algoritmo, un flujo o un proceso.

Wireframe → Guía visual que representa el esqueleto o estructura de un sitio web.

ANEXOS

A. SCRIPT ANÁLISIS

A continuación se muestran algunos fragmentos del código del script encargado del análisis de código de la herramienta CheckOode. Se ha tomado la decisión de no incluir, en este anexo, el código correspondiente a la interfaz de usuario, debido al elevado número de líneas de código del que está compuesto. Por otro lado, dicho código correspondiente a la interfaz de usuario ha sido entregado al tutor del trabajo de fin de grado para que éste pueda evaluarlo.

```
##########################################################################################
# Función encargada de la búsqueda del Makefile y de comprobar que este sea correcto
##########################################################################################

analyseMakefile (){

    RULES="all clean doc"
    COMPILE="Wall ansi pedantic"

    if [ -f "$1" ]
    then
        contador=0

        for r in $RULES
        do
        LINEA=$(grep -o '^'$r':' $1)
        if [ -z $LINEA ]
        then
            if [  $contador -gt 0 ]
            then
                echo ",{\"value\":\"missing $r rule\"}" >> analisis.txt
            else
                echo "{\"value\":\"missing $r rule\"}" >> analisis.txt
            fi
            contador=$(( $contador + 1 ))
        fi
        done

        for opt in $COMPILE
        do
        LINEA=$(grep -o $opt $1)
        if [ -z $LINEA ]
        then
            if [  $contador -gt 0 ]
            then
                echo ",{\"value\":\"missing $opt option\"}" >> analisis.txt
            else
                echo "{\"value\":\"missing $opt option\"}" >> analisis.txt
            fi
        fi
        done
    else
        echo "{\"value\":\"There is no makefile.\"}" >> analisis.txt
    fi

}
```

Figura A.1: *Función encargada del análisis del makefile*

```
#########################################################################
# Función encargada de la búsqueda de los autores del Proyecto
#########################################################################

analyseAuthors (){

    for i in $1
    do
        egrep -i 'author|autor' $i \
        | tr ":" " " | tr -d [=*=] | tr -s " " | tr A-Z a-z \
        | cut -d' ' -f3-20 | sed -e 's/[[:space:]]*$//' \
        | sort -u >> autores.txt

    done

}
```

Figura A.2: *Función encargada del análisis de los autores*

```
#########################################################################################
# Función encargada de comprobar la existencia de números mágicos
#########################################################################################

magicNumbers (){

    for i in $1
    do

        egrep --color -n '(\[[0-9]+\])|([!<>=]?=\s*[1-9][0-9|\.]*)' $1 > magicNum.txt
        tr -d '{}";/* ' < magicNum.txt > magicNum2.txt
        tr ':' ' ' < magicNum2.txt > magicNum3.txt

    done

}
```

Figura A.3: *Función encargada de la detección de números mágicos*

```
#########################################################################################
# Función encargada de la ejecución del análisis de la herramienta pmccabe
#########################################################################################

mcabe (){


    for i in $1
    do
        pmccabe "$i" > mcabe.dat
    done

    ls *dat > datos.txt
    DATOS=$(cut -f 1 -d "." datos.txt)

    for i in $DATOS
    do
        awk '{print $7" "$5"    "$4"    "$3"    "$2"    "$1}' "$i".dat > "$i"_mCabe.txt
    done

}
```

Figura A.4: *Función encargada de la ejecución de la herramienta Pmccabe*

```
#############################################################################################
# Función encargada de la ejecución del análisis de la herramienta ctags
#############################################################################################

ctagsfunc (){

    ctags --fields=afmikKlnsStz -f tags.txt $1
    sed -i".bak" '/!_TAG/d' tags.txt
    awk -F '\t' '$4 == "kind:macro" {print $1,$5}' tags.txt > define_tags.txt
    awk -F '\t' '$4 == "kind:function" {print $1"   "$7}' tags.txt > function_tags.txt

}
```

Figura A.5: *Función encargada de la ejecución de la herramienta Ctags*

```
#############################################################################################
# Función encargada de la búsqueda del uso de recursos
#############################################################################################

check_resources (){

    for i in $1
    do
        awk '/fopen/{ gsub(/ /, "", $0);
              printf "%-20s:%-5s: %-7s: %-s\n", FILENAME, FNR, "open", $0}
            /fclose/{gsub(/ /, "", $0);
              printf "%-20s:%-5s: %-7s: %-s\n", FILENAME, FNR, "close", $0}
            /malloc|realloc|calloc/{gsub(/ /, "", $0);
              printf "%-20s:%-5s: %-7s: %-s\n", FILENAME, FNR, "alloc", $0}
            /free/{gsub(/ /, "", $0);
              printf "%-20s:%-5s: %-7s: %-s\n", FILENAME, FNR, "free", $0}' "$i" > resources.txt
        tr -d '"*' < resources.txt > resources2.txt
        tr ':' ' ' < resources2.txt > resources3.txt
    done

}
```

Figura A.6: *Función encargada de la detección de recursos*

```
echo "]," >> analisis.txt
echo "\"numMagicos\":[" >> analisis.txt
echo "\033[0;32mBuscando Numeros Magicos...\033[1;m"

magicNumbers proyectos/$PROYECTO/$j

if [ -f magicNum2.txt -a ! -s magicNum2.txt ]
then
    echo "{\"linea\":\"NO HAY NÚMEROS MÁGICOS\", \"codigo\":\"-\"}" >> analisis.txt
else
    flag=0

    while read line
    do
        contNum=0
        for h in $line
        do
            if [ $contNum -gt 0 ]
            then
                echo "\"codigo\":\"$h\"" >> analisis.txt
            else
                if [ $flag -gt 0 ]
                then
                    echo ",{\"linea\":\"$h\"," >> analisis.txt
                    contNum=1
                else
                    echo "{\"linea\":\"$h\"," >> analisis.txt
                    contNum=1
                    flag=1
                fi
            fi
        done
        contNum=0
        echo "}" >> analisis.txt
    done < magicNum3.txt

fi
```

Figura A.7: *Ejemplo de gestión de ficheros e inserción de datos en el archivo JSON*

B. MAQUETAS INTERFAZ USUARIO

En este anexo se encuentra una galería compuesta por las maquetas realizadas para la interfaz de usuario de la herramienta CheckOode. Todas ellas son una primera versión de lo que se esperaba implementar finalmente para la interfaz de usuario y han sido la base de apoyo para la realización de la misma.

En cada una de las imágenes que se pueden encontrar a continuación hay una etiqueta que especifica que es lo que contiene cada una de las maquetas de las pantallas.

Figura B.1: *Maqueta Pantalla Principal*

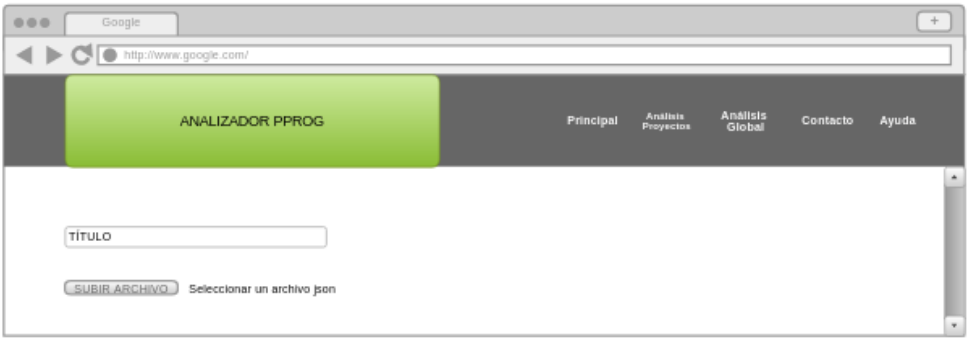


Figura B.2: *Maqueta Común para Selección Fichero Resultados*

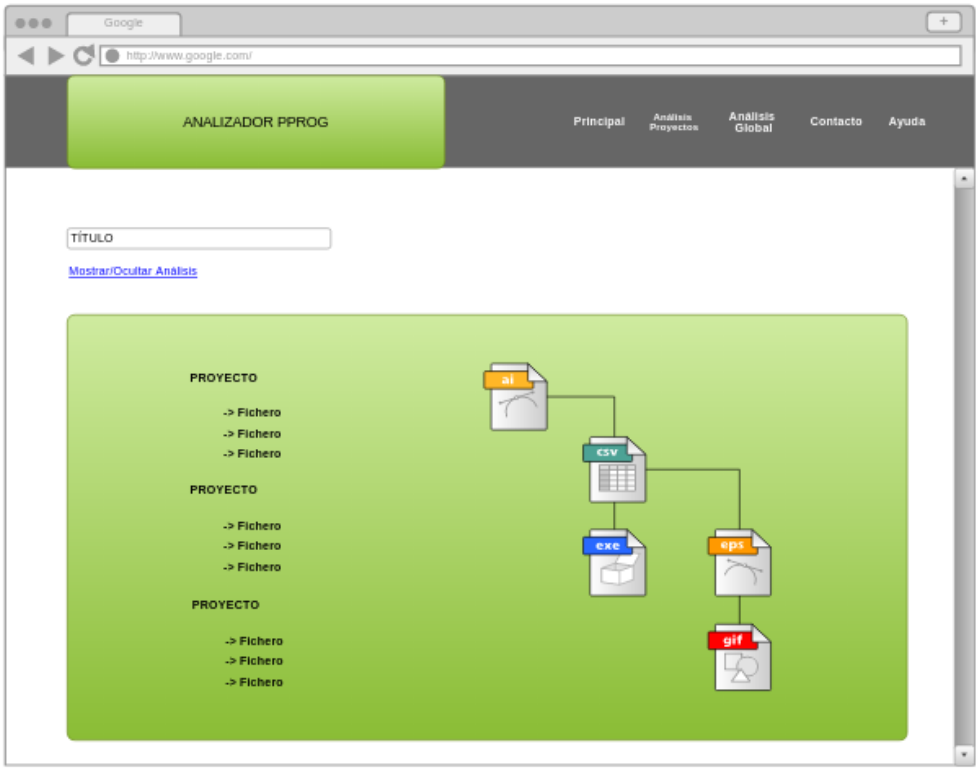


Figura B.3: *Maqueta Pantalla Análisis de Estructura Proyectos*

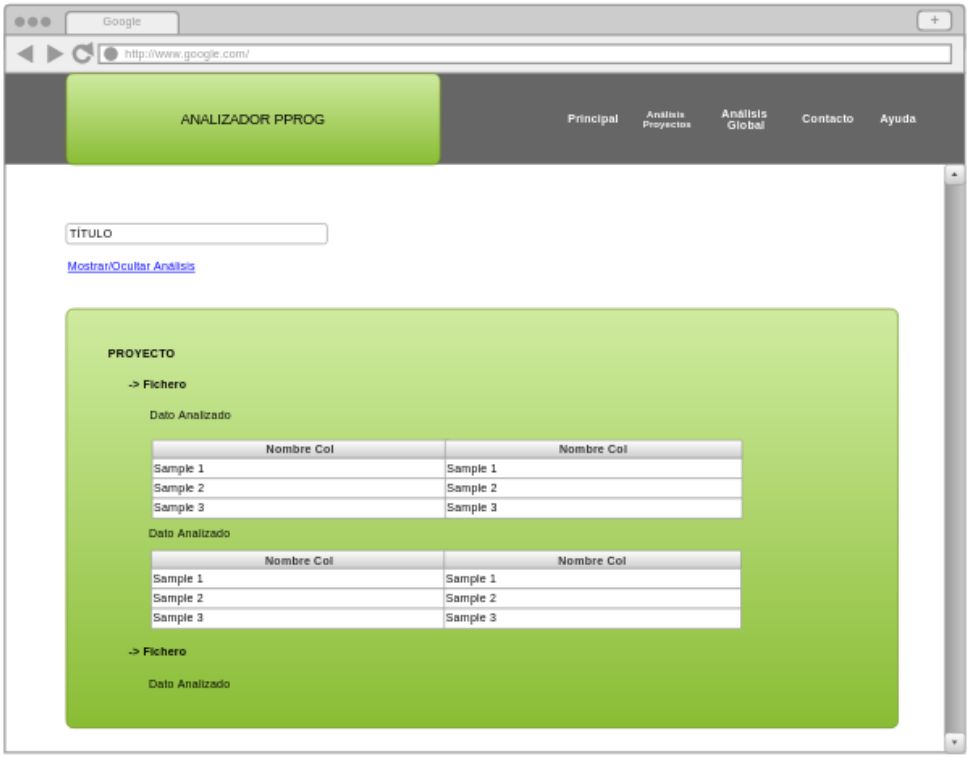


Figura B.4: *Maqueta Pantalla Análisis Tablas Proyectos*

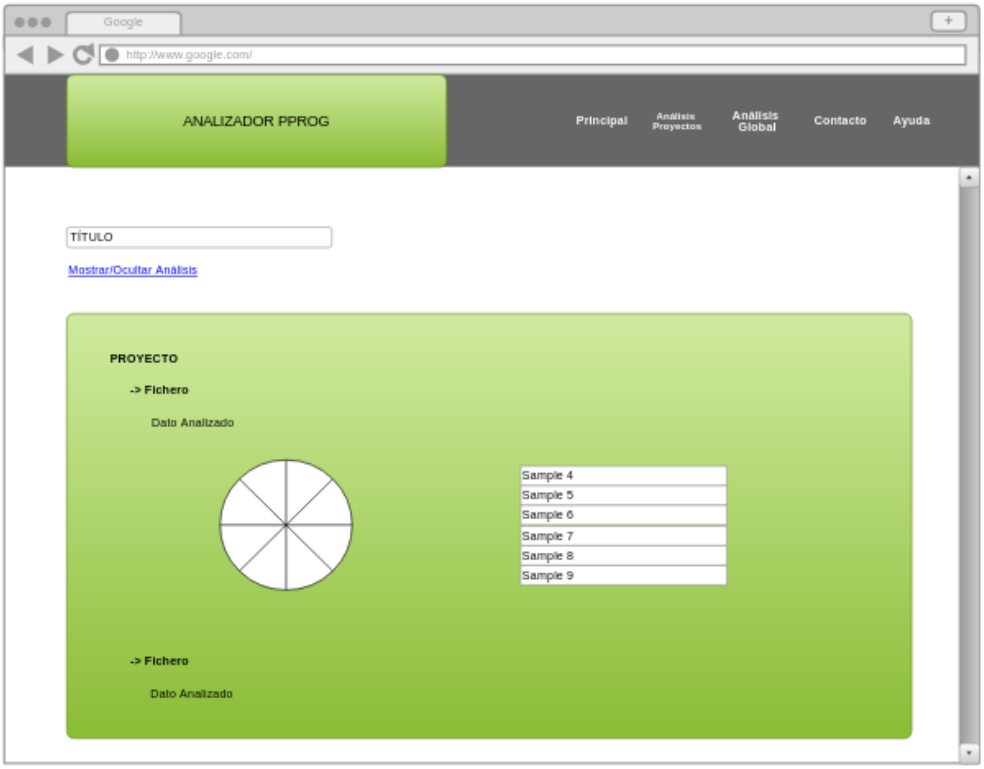


Figura B.5: *Maqueta Pantalla Análisis Gráficos Proyectos*

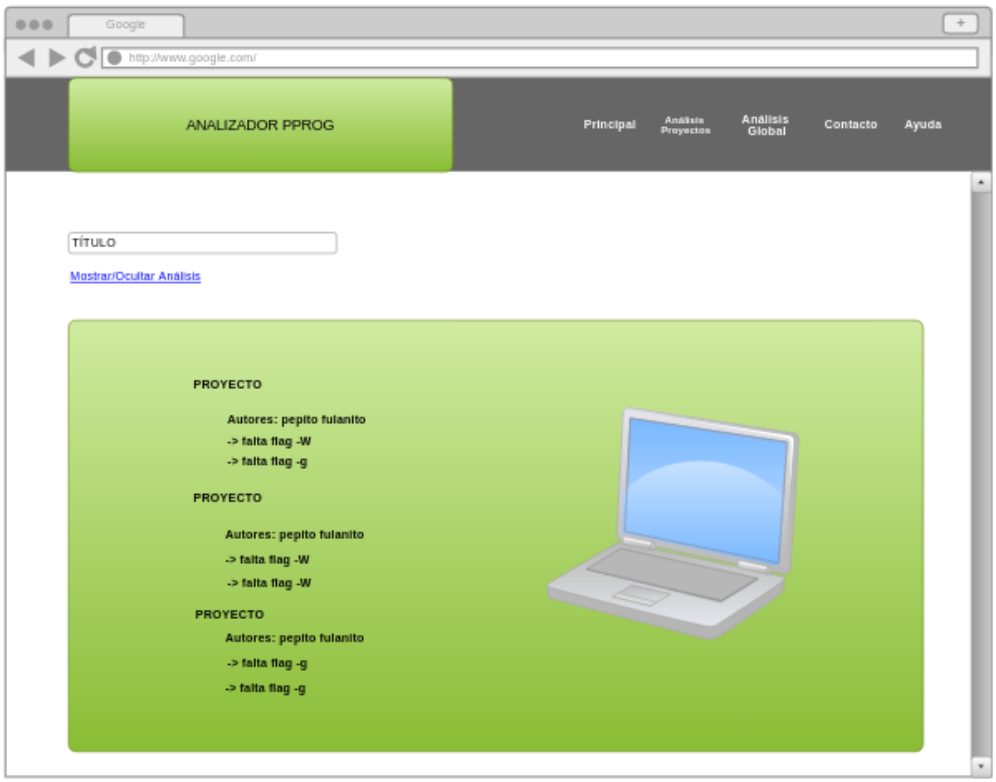


Figura B.6: *Maqueta Pantalla Análisis Makefile y Autores Proyectos*

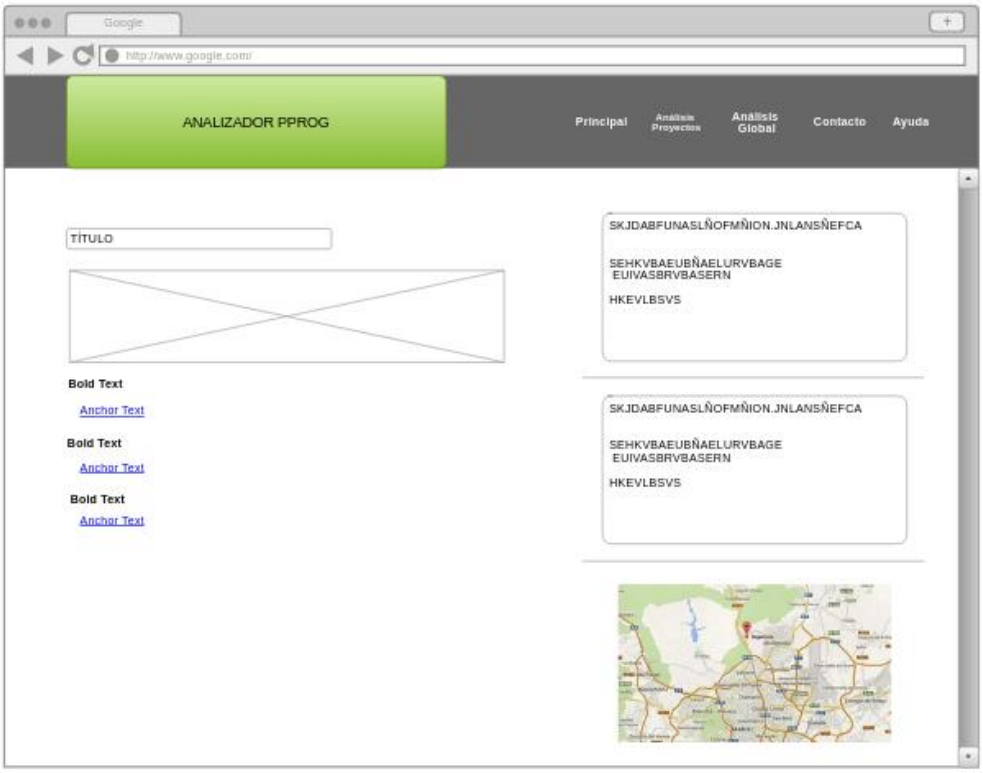


Figura B.7: *Maqueta Pantalla Contacto*

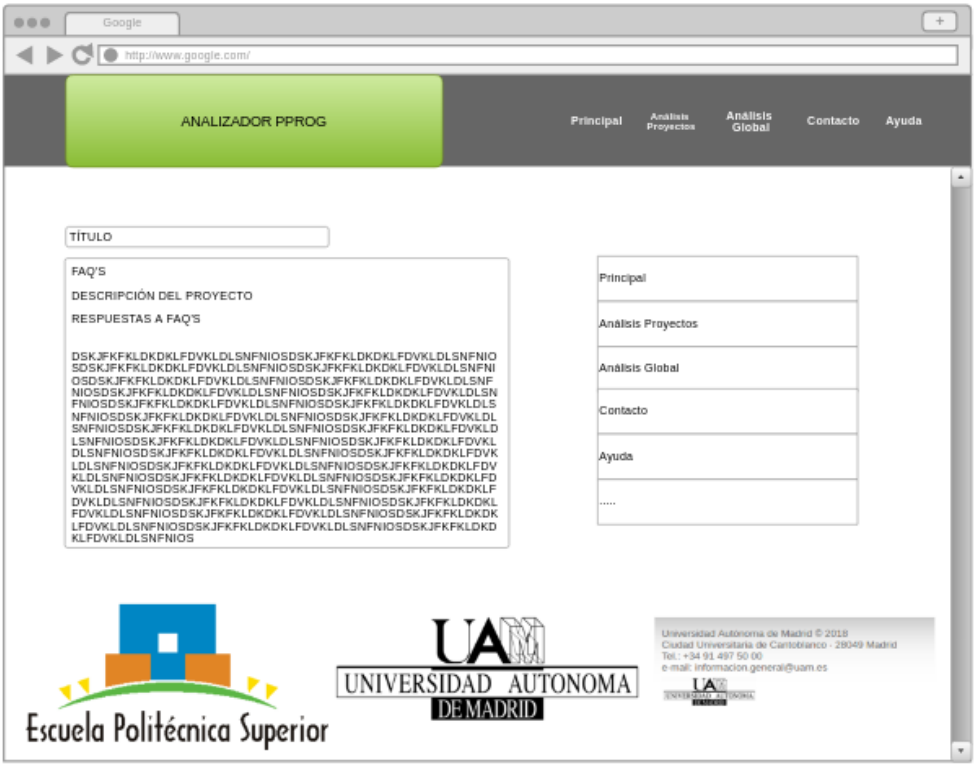


Figura B.8: *Maqueta Pantalla Ayuda*

C. IMÁGENES INTERFAZ WEB

A continuación, se muestran las capturas de pantalla correspondientes a todas las pantallas que integran la interfaz de usuario de ***CheckOode.***

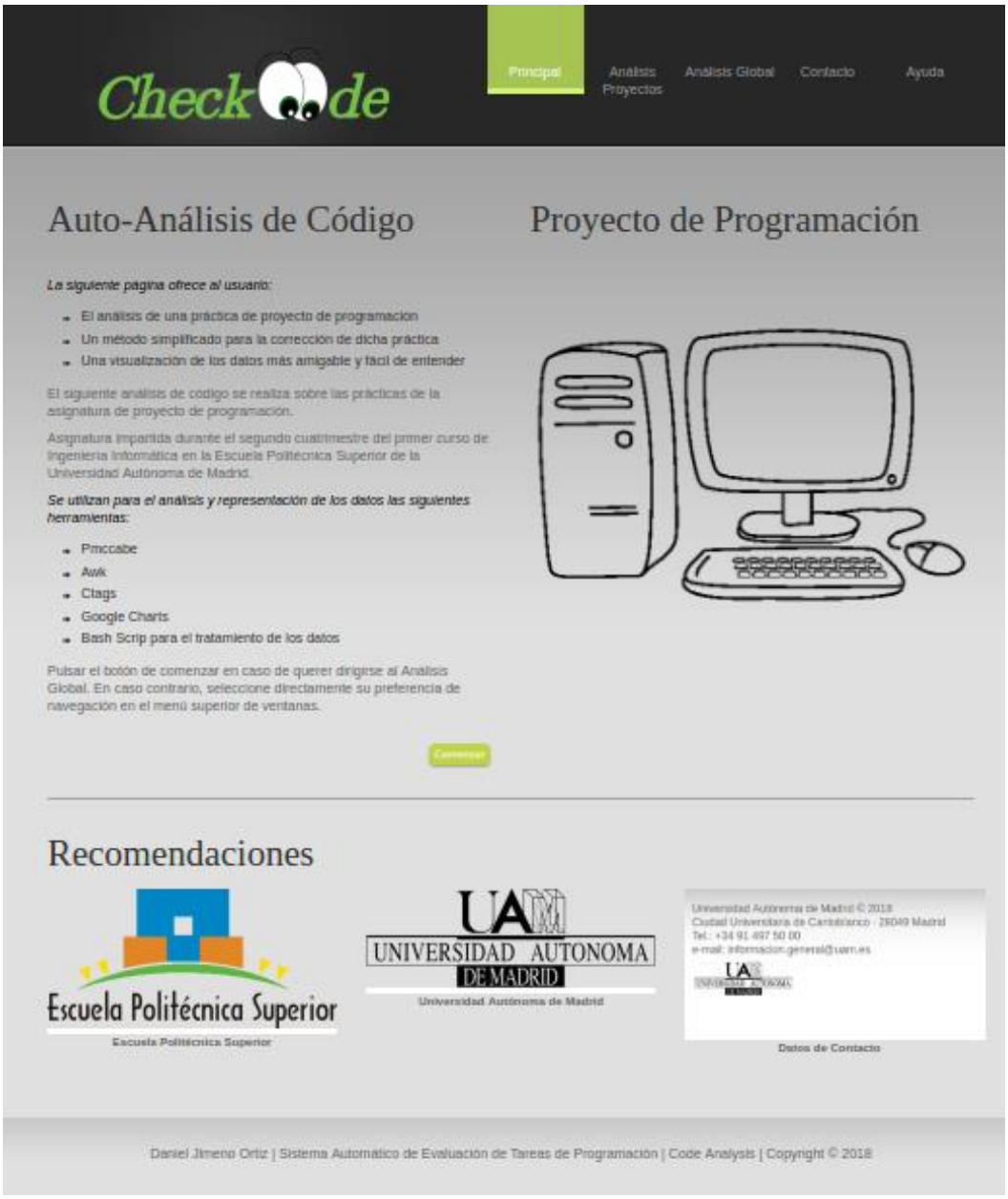


Figura C.1: *Página Principal*

Figura C.2: *Pantalla Análisis Proyectos*

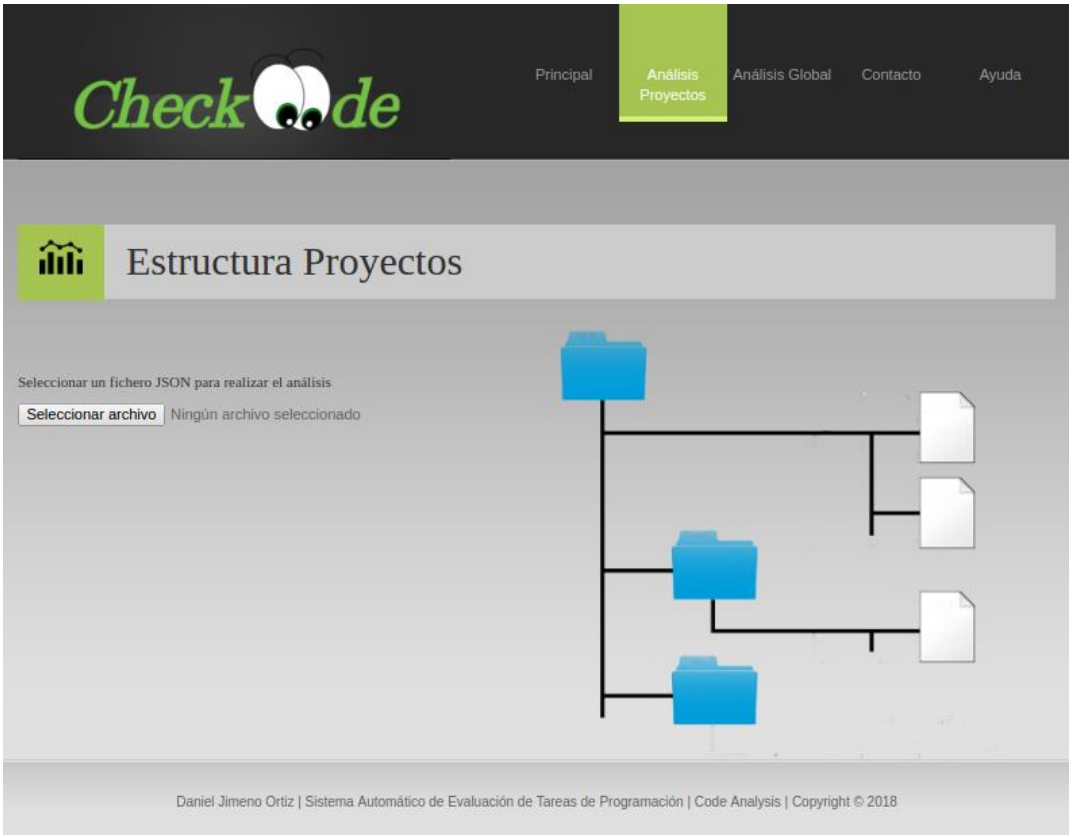


Figura C.3: *Pantalla Selección Archivo Análisis Estructura de Proyectos*

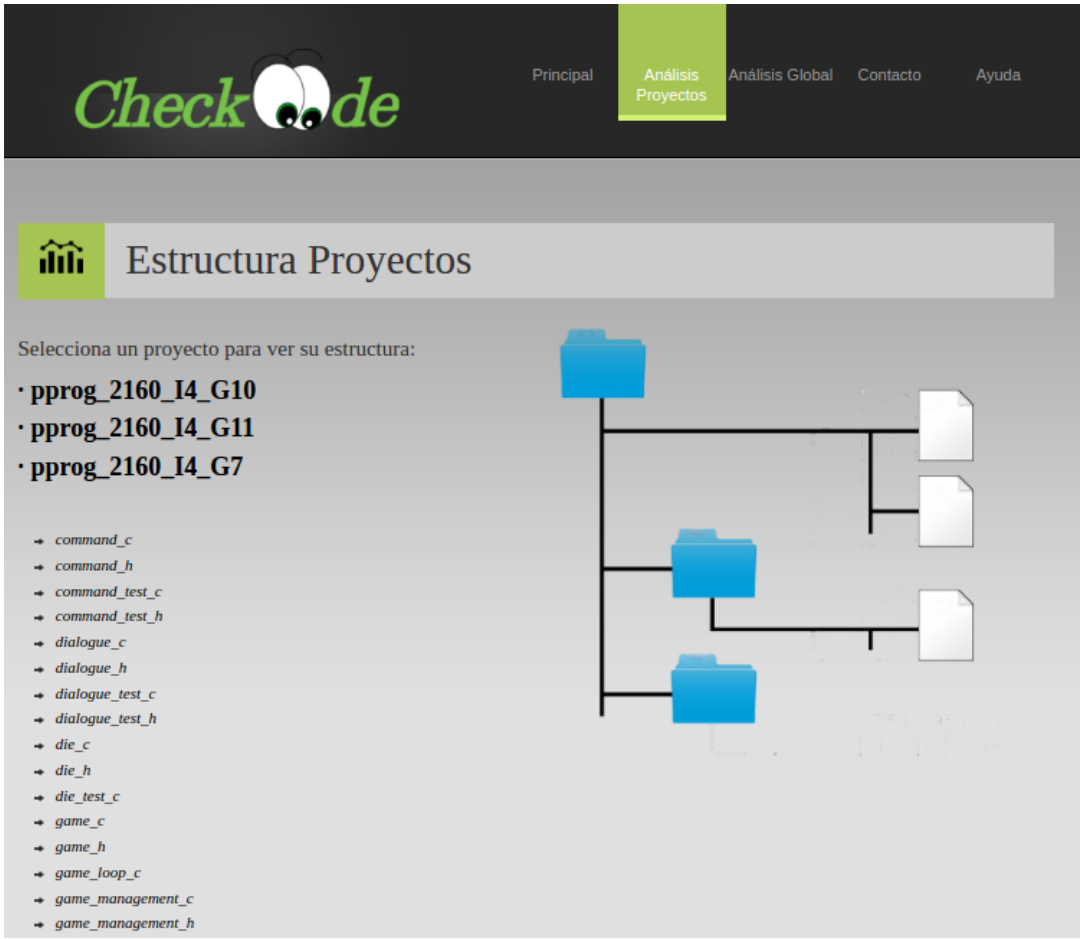


Figura C.4: *Pantalla Resultados Análisis Estructura de Proyectos*

Figura C.5: *Pantalla Selección Archivo Análisis Tablas de Proyectos*

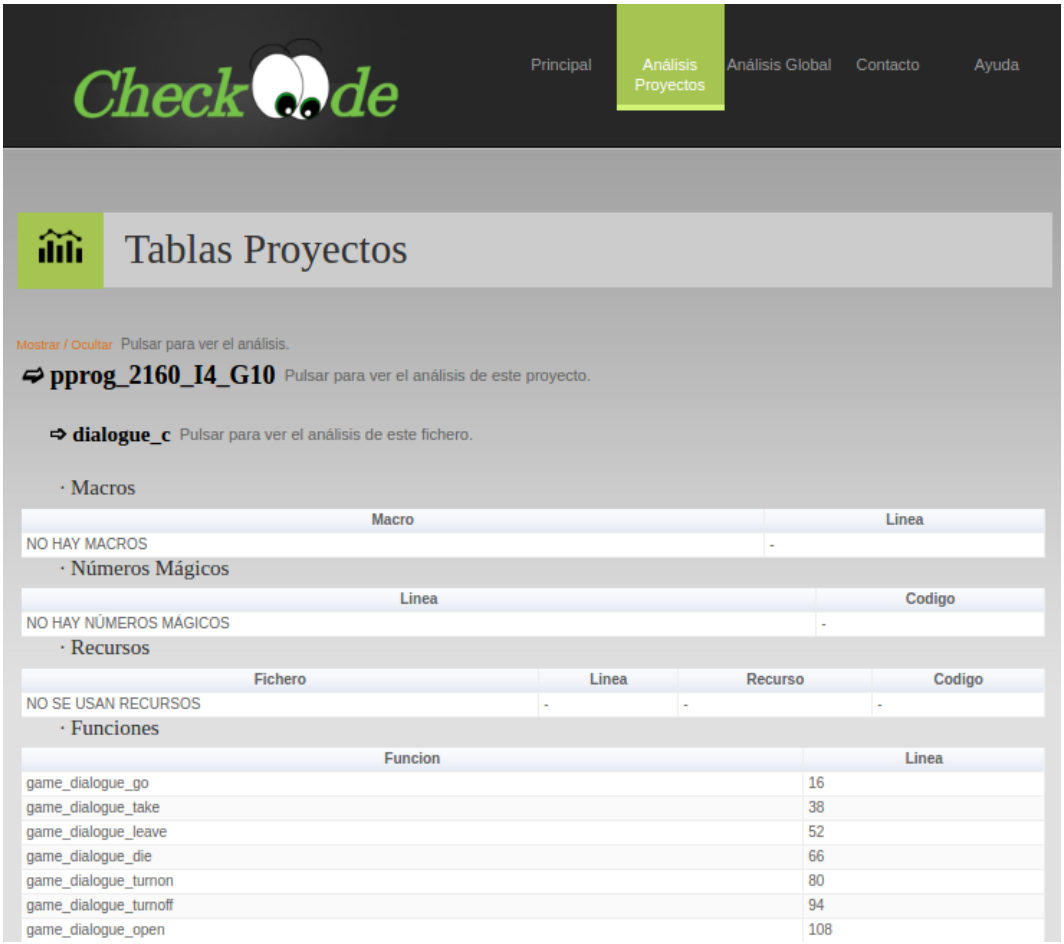


Figura C.6: *Pantalla Resultados Análisis Tablas Proyectos*

Figura C.7: *Pantalla Selección Archivo Gráficos de Proyectos*

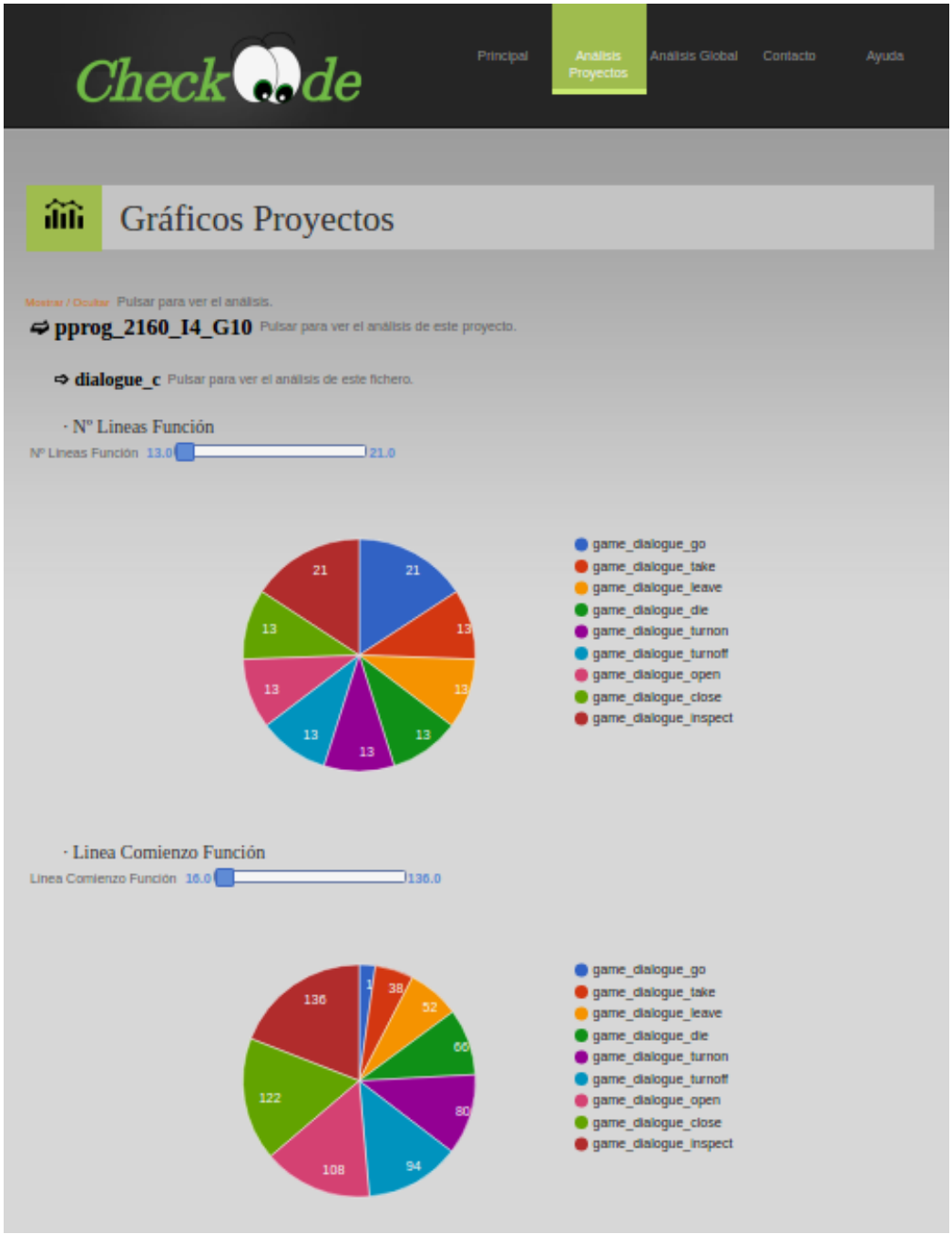


Figura C.8: *Pantalla Resultados Gráficos de Proyectos*

Figura C.9: *Pantalla Selección Archivo Análisis Makefile y Autores de Proyectos*

Figura C.10: *Pantalla Resultados Makefile y Autores Análisis de Proyectos*

Figura C.11: *Pantalla Análisis Global*

Figura C.12: *Pantalla Selección Archivo Análisis Tablas Global*

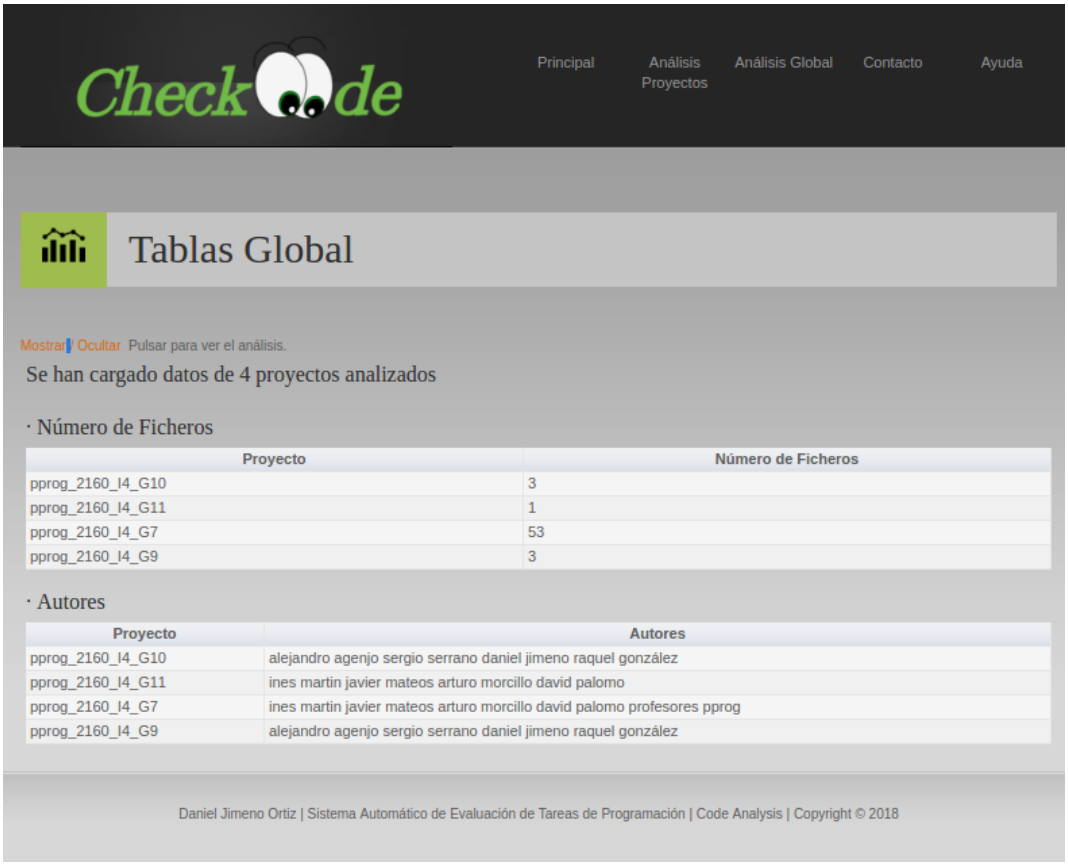


Figura C.13: *Pantalla Resultados Análisis Tablas Global*

Figura C.14: *Pantalla Selección Archivos Gráficos Global*

Figura C.15: *Pantalla Resultados Análisis Gráficos Global*

Figura C.16: *Pantalla Contacto*

Figura C.17: *Pantalla Ayuda*

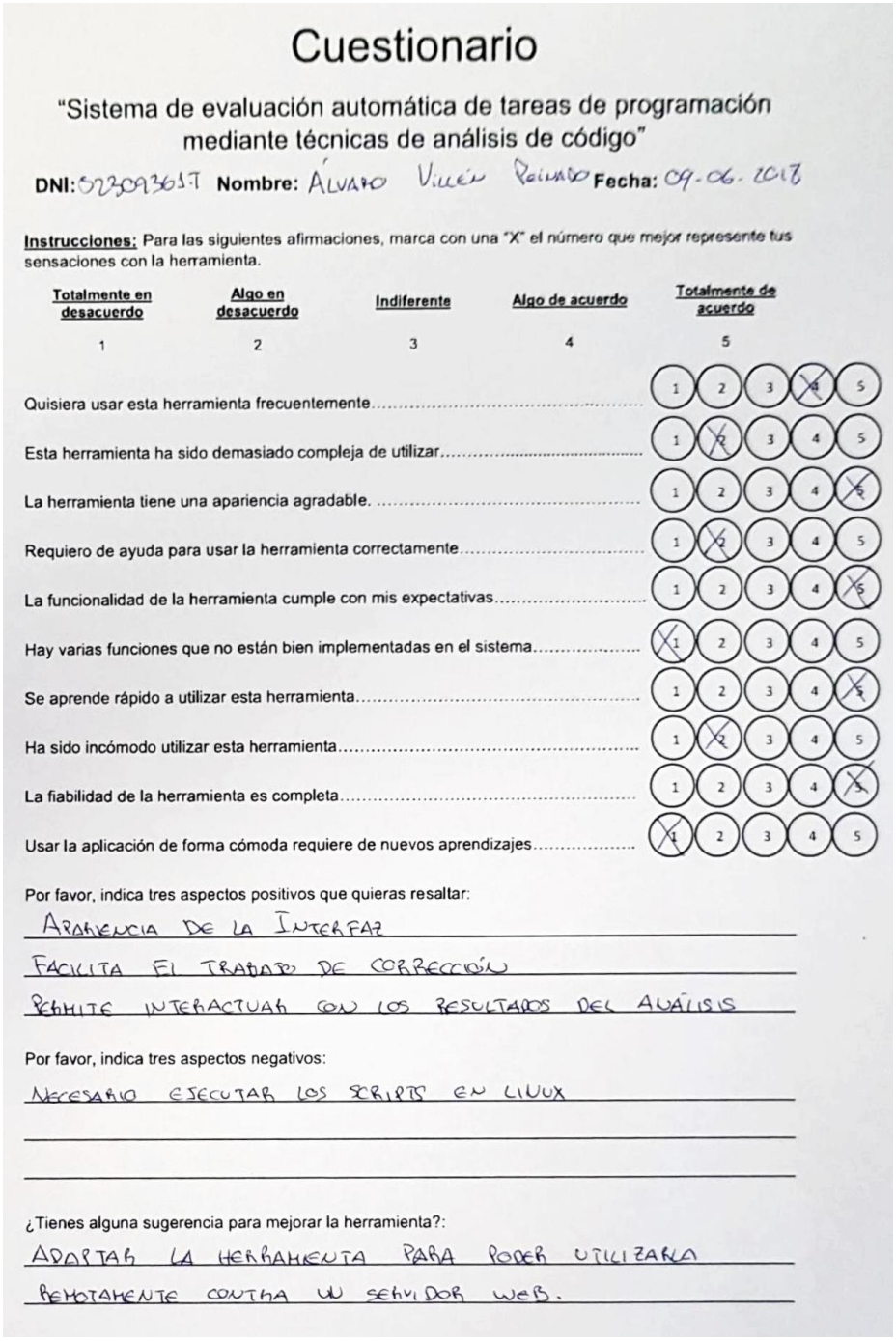
D. LOGOTIPO CHECKOODE

A continuación se expone el logotipo del nombre de la herramienta que ha sido diseñado y creado por el desarrollador del proyecto utilizando la herramienta Cacoo.

Figura D.1: *Logotipo Herramienta*

E. ENCUESTAS

A continuación, se muestran las encuestas realizadas por los diferentes usuarios que han hecho uso de la herramienta con el fin de cumplimentar las encuestas y probar la herramienta.



Cuestionario

"Sistema de evaluación automática de tareas de programación mediante técnicas de análisis de código"

DNI: 07230936J **Nombre:** ÁLVARO VILLÉN PALMERO **Fecha:** 09-06-2017

Instrucciones: Para las siguientes afirmaciones, marca con una *"X" el número que mejor represente tus* sensaciones con la herramienta.

Totalmente en desacuerdo	Algo en desacuerdo	Indiferente	Algo de acuerdo	Totalmente de acuerdo
1	2	3	4	5

Quisiera usar esta herramienta frecuentemente	4
Esta herramienta ha sido demasiado compleja de utilizar	2
La herramienta tiene una apariencia agradable.	5
Requiero de ayuda para usar la herramienta correctamente	2
La funcionalidad de la herramienta cumple con mis expectativas	5
Hay varias funciones que no están bien implementadas en el sistema	1
Se aprende rápido a utilizar esta herramienta	5
Ha sido incómodo utilizar esta herramienta	2
La fiabilidad de la herramienta es completa	5
Usar la aplicación de forma cómoda requiere de nuevos aprendizajes	1

Por favor, indica tres aspectos positivos que quieras resaltar:

APARIENCIA DE LA INTERFAZ

FACILITA EL TRABAJO DE CORRECCIÓN

PERMITE INTERACTUAR CON LOS RESULTADOS DEL ANÁLISIS

Por favor, indica tres aspectos negativos:

NECESARIO EJECUTAR LOS SCRIPTS EN LINUX

¿Tienes alguna sugerencia para mejorar la herramienta?:

ADAPTAR LA HERRAMIENTA PARA PODER UTILIZARLA REMOTAMENTE CONTRA UN SERVIDOR WEB.

Figura E.1: *Encuesta Nº 1*

Cuestionario

"Sistema de evaluación automática de tareas de programación mediante técnicas de análisis de código"

DNI: 71715512V **Nombre:** Borja González Farías **Fecha:** 09/06/2018

Instrucciones: Para las siguientes afirmaciones, marca con una "X" el número que mejor represente tus sensaciones con la herramienta.

Totalmente en desacuerdo	Algo en desacuerdo	Indiferente	Algo de acuerdo	Totalmente de acuerdo
1	2	3	4	5

Quisiera usar esta herramienta frecuentemente	5
Esta herramienta ha sido demasiado compleja de utilizar	2
La herramienta tiene una apariencia agradable.	4
Requiero de ayuda para usar la herramienta correctamente	1
La funcionalidad de la herramienta cumple con mis expectativas	4
Hay varias funciones que no están bien implementadas en el sistema	1
Se aprende rápido a utilizar esta herramienta	5
Ha sido incómodo utilizar esta herramienta	2
La fiabilidad de la herramienta es completa	5
Usar la aplicación de forma cómoda requiere de nuevos aprendizajes	1

Por favor, indica tres aspectos positivos que quieras resaltar:

1) Es cómoda de utilizar
2) Ahorra tiempo para corregir entregas
3) El flujo entre las diferentes páginas resulta agradable

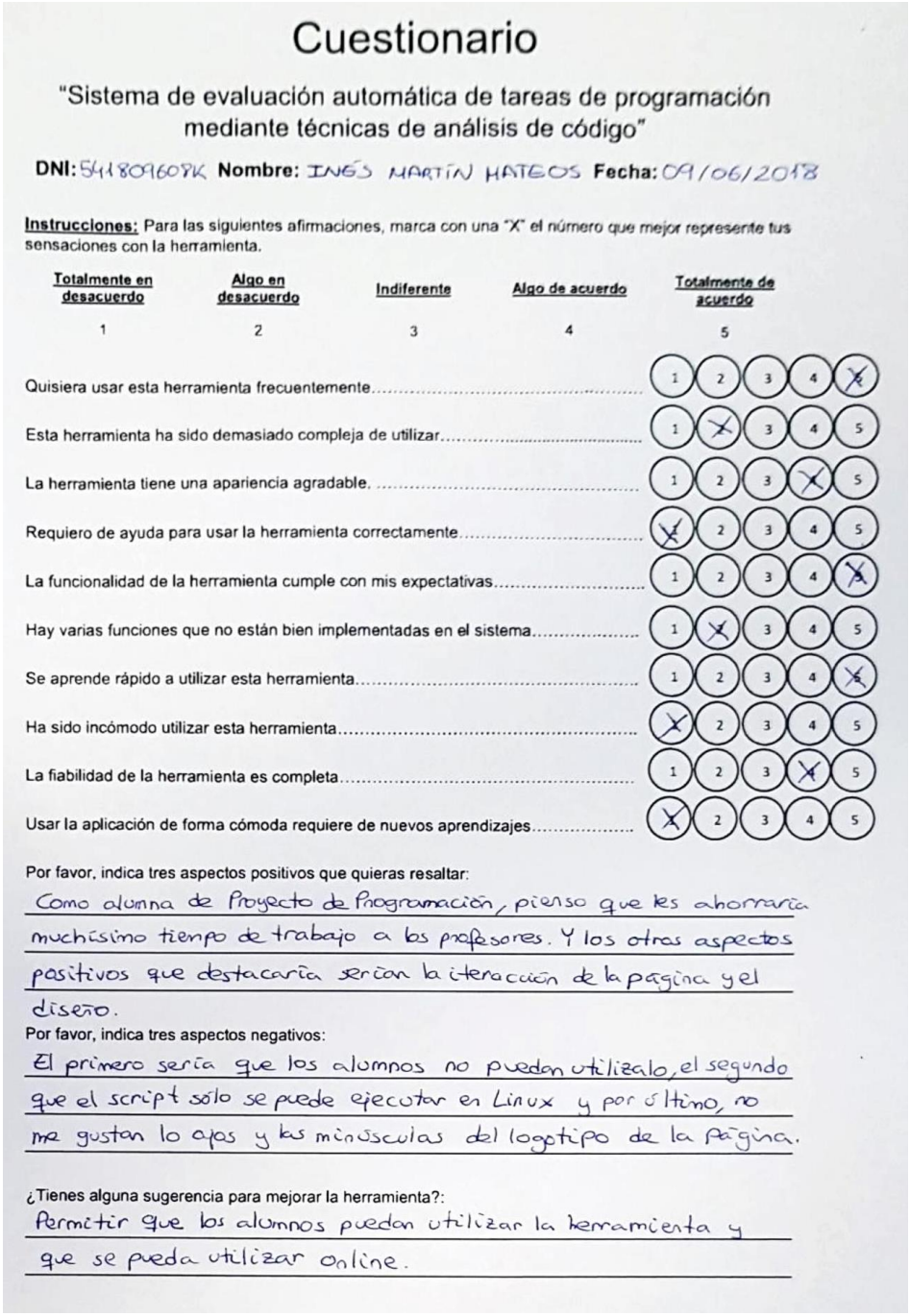
Por favor, indica tres aspectos negativos:

1) El script carece de interoperabilidad
2) Sólo limitada al uso de profesores
3) No se puede utilizar online.

¿Tienes alguna sugerencia para mejorar la herramienta?:

Que la aplicación se utilice en diferentes sistemas operativos.

Figura E.2: *Encuesta Nº 2*



Cuestionario

"Sistema de evaluación automática de tareas de programación mediante técnicas de análisis de código"

DNI: 54180960BK **Nombre:** INÉS MARTÍN MATEOS **Fecha:** 09/06/2018

Instrucciones: Para las siguientes afirmaciones, marca con una "X" el número que mejor represente tus sensaciones con la herramienta.

Totalmente en desacuerdo	Algo en desacuerdo	Indiferente	Algo de acuerdo	Totalmente de acuerdo
1	2	3	4	5

	1	2	3	4	5
Quisiera usar esta herramienta frecuentemente					X
Esta herramienta ha sido demasiado compleja de utilizar		X			
La herramienta tiene una apariencia agradable.				X	
Requiero de ayuda para usar la herramienta correctamente	X				
La funcionalidad de la herramienta cumple con mis expectativas					X
Hay varias funciones que no están bien implementadas en el sistema		X			
Se aprende rápido a utilizar esta herramienta					X
Ha sido incómodo utilizar esta herramienta	X				
La fiabilidad de la herramienta es completa				X	
Usar la aplicación de forma cómoda requiere de nuevos aprendizajes	X				

Por favor, indica tres aspectos positivos que quieras resaltar:

Como alumna de Proyecto de Programación, pienso que les ahorraría muchísimo tiempo de trabajo a los profesores. Y los otros aspectos positivos que destacaría serían la interacción de la página y el diseño.

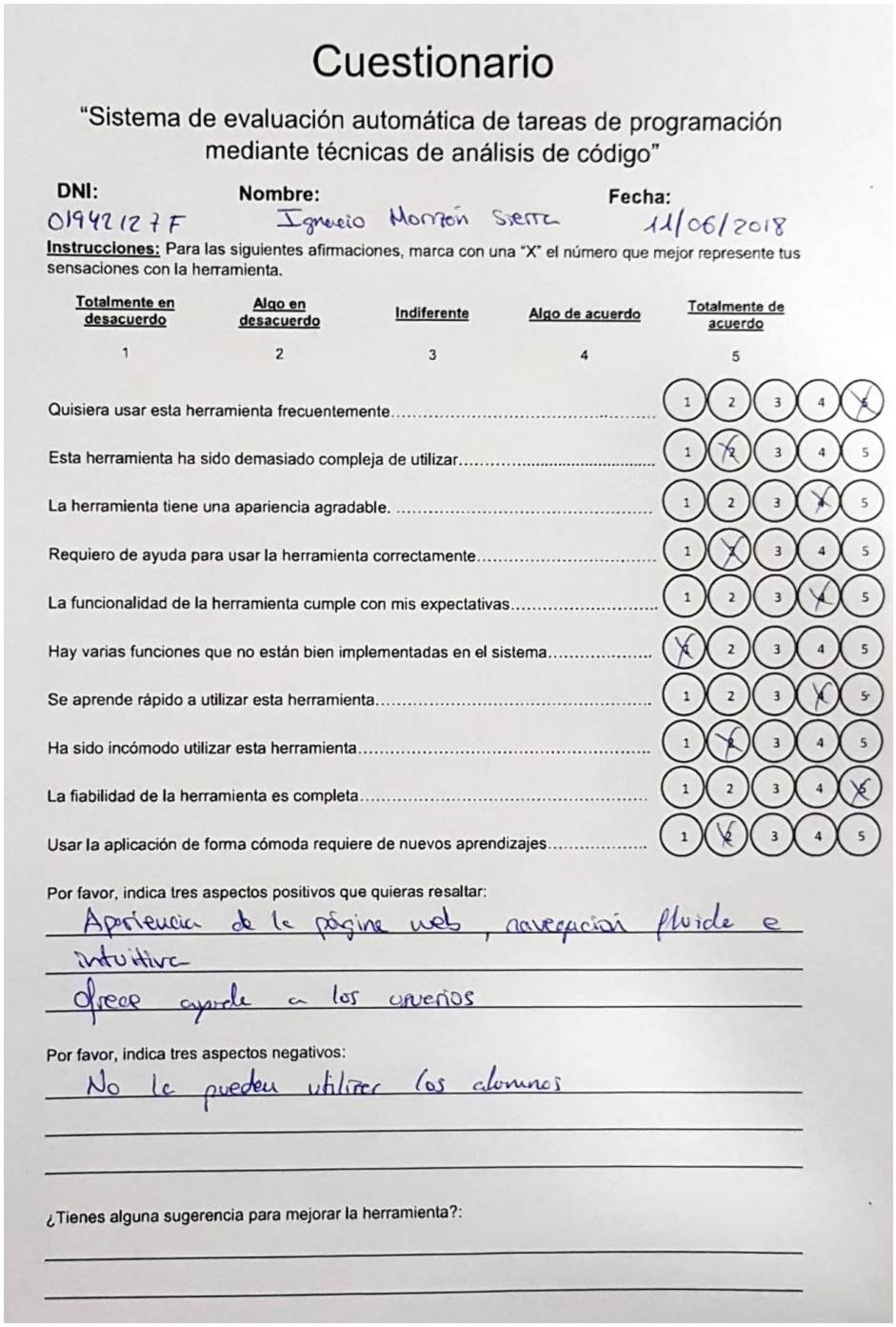
Por favor, indica tres aspectos negativos:

El primero sería que los alumnos no puedan utilizarlo, el segundo que el script sólo se puede ejecutar en Linux y por último, no me gustan lo ojos y las minúsculas del logotipo de la página.

¿Tienes alguna sugerencia para mejorar la herramienta?:

Permitir que los alumnos puedan utilizar la herramienta y que se pueda utilizar online.

Figura E.3: *Encuesta N º 3*



Cuestionario

"Sistema de evaluación automática de tareas de programación mediante técnicas de análisis de código"

DNI: 01942127F **Nombre:** Ignacio Morrón Sierra **Fecha:** 11/06/2018

Instrucciones: Para las siguientes afirmaciones, marca con una "X" el número que mejor represente tus sensaciones con la herramienta.

Totalmente en desacuerdo	Algo en desacuerdo	Indiferente	Algo de acuerdo	Totalmente de acuerdo
1	2	3	4	5

Quisiera usar esta herramienta frecuentemente	5
Esta herramienta ha sido demasiado compleja de utilizar	2
La herramienta tiene una apariencia agradable.	4
Requiero de ayuda para usar la herramienta correctamente	2
La funcionalidad de la herramienta cumple con mis expectativas	4
Hay varias funciones que no están bien implementadas en el sistema	1
Se aprende rápido a utilizar esta herramienta	4
Ha sido incómodo utilizar esta herramienta	2
La fiabilidad de la herramienta es completa	5
Usar la aplicación de forma cómoda requiere de nuevos aprendizajes	2

Por favor, indica tres aspectos positivos que quieras resaltar:

Apariencia de la página web, navegación fluida e intuitiva

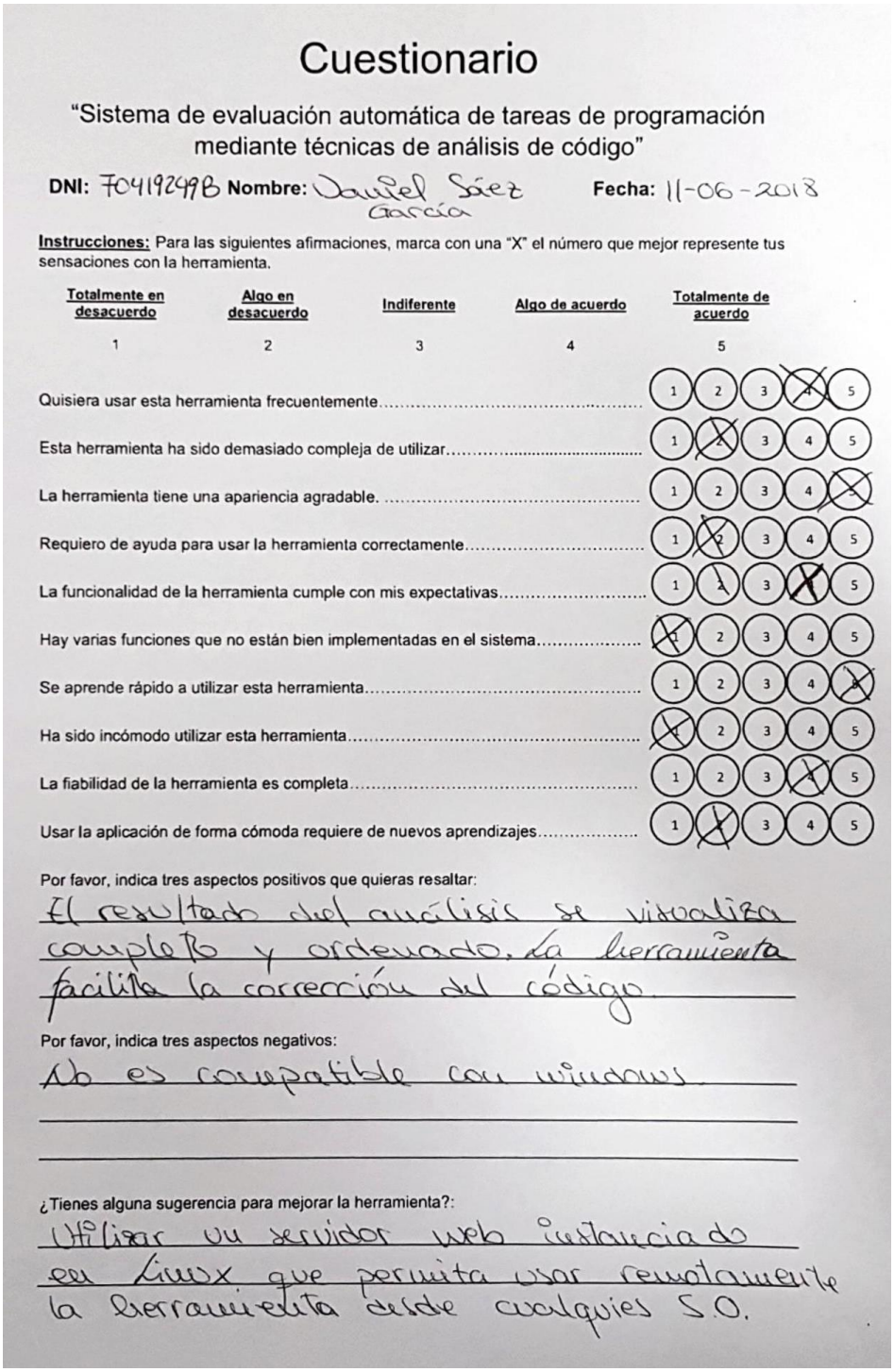
Ofrece ayuda a los usuarios

Por favor, indica tres aspectos negativos:

No le pueden utilizar los alumnos

¿Tienes alguna sugerencia para mejorar la herramienta?:

Figura E.4: *Encuesta N º 4*



Cuestionario

"Sistema de evaluación automática de tareas de programación mediante técnicas de análisis de código"

DNI: 70419249B **Nombre:** Daniel Sáez García **Fecha:** 11-06-2018

Instrucciones: Para las siguientes afirmaciones, marca con una "X" el número que mejor represente tus sensaciones con la herramienta.

Totalmente en desacuerdo	Algo en desacuerdo	Indiferente	Algo de acuerdo	Totalmente de acuerdo
1	2	3	4	5

Quisiera usar esta herramienta frecuentemente	5
Esta herramienta ha sido demasiado compleja de utilizar	2
La herramienta tiene una apariencia agradable.	5
Requiero de ayuda para usar la herramienta correctamente	2
La funcionalidad de la herramienta cumple con mis expectativas	4
Hay varias funciones que no están bien implementadas en el sistema	1
Se aprende rápido a utilizar esta herramienta	5
Ha sido incómodo utilizar esta herramienta	1
La fiabilidad de la herramienta es completa	4
Usar la aplicación de forma cómoda requiere de nuevos aprendizajes	2

Por favor, indica tres aspectos positivos que quieras resaltar:

El resultado del análisis se visualiza completo y ordenado. La herramienta facilita la corrección del código.

Por favor, indica tres aspectos negativos:

No es compatible con windows.

¿Tienes alguna sugerencia para mejorar la herramienta?:

Utilizar un servidor web instanciado en Linux que permita usar remotamente la herramienta desde cualquier S.O.

Figura E.5: *Encuesta N º 5*